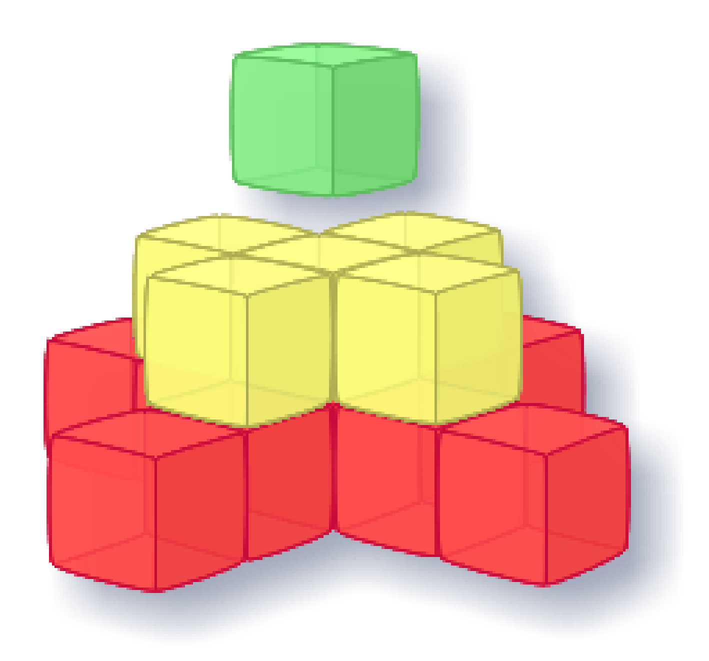
Dipl. Ing. Toralf Riedel
Dipl. Ing. Päd. Alexander Huwaldt

Benutzerhandbuch

SiSy® AVR Mikrocontroller

gültig ab SiSy-Version 2.18



5. Auflage: Oktober 2010

www.laser-co.de
www.myavr.de
info@myavr.de
Tel: ++49 (0) 3585 470 222
Fax: ++49 (0) 3585 470 233

Inhalt

1 Einleitung

Sie haben eine Ausgabe des Modellierungswerkzeuges Simple System, kurz SiSy, erworben. Bevor auf die verschiedenen Funktionen des Programms eingegangen wird, noch einige Worte zum vorliegenden Handbuch. Mit Hilfe des Handbuchs werden dem Nutzer die Grundlagen der Bedienung von SiSy erläutert. Der Inhalt, die Gestalt und die Regeln der Modelle werden nur am Rand betrachtet. Das genaue Vorgehen für die Programmierung eines Mikroprozessors wird an einem Beispiel ausführlich beschrieben. Auf die Grundlagen der Mikroprozessorprogrammierung wird im Rahmen dieses Handbuches nicht eingegangen. Dazu dienen die myAVR Lehrbücher.

Dem Nutzer wird in diesem Handbuch der Einstieg in das Programm erleichtert und die umfangreichen Funktionen von SiSy kurz und verständlich beschrieben. Bei der Arbeit mit SiSy erstellt der Anwender Modelle in Form von Diagrammen und in ihnen enthaltene Symbole. Die Grundlagen der Entstehung und Bearbeitung solcher Diagramme sind Gegenstand der Betrachtung dieses Handbuchs.

Folgende Darstellungs- und Gestaltungsmittel sind für den Nutzer bei der Arbeit mit SiSy besonders wichtig:

- die Diagramme als Fenster zur Ansicht und Bearbeitung von Modellen;
- der Navigator als Fenster zur Steuerung und Bewegung in Modellen;
- der Assistent mit Hilfestellungen zum jeweils geöffneten Diagramm und mit Diagrammvorlagen (wenn vorhanden);
- die Menüs und Schalter für Befehle an Navigator, Diagramm und Objekt im Kontext mit der Modellierung.

Zu den Bezeichnungen im Text:

- Falls ein Menübefehl nur über Untermenüs zu erreichen ist, werden die einzelnen Menübezeichnungen kursiv geschrieben und durch Schrägstriche voneinander getrennt.
  Beispiel: Menü *Hilfe/über SiSy*

- Titel von Dialogboxen, Schaltflächen und Menüpunkten werden in Anführungszeichen gesetzt.
  Beispiel: Dialogbox „Definition“, Schaltfläche „OK“

2 Vorbereitung

In diesem Kapitel werden Sie über notwendige Schritte zur Beschaffung, Installation, Konfiguration und Aufbau einer funktionsfähigen Entwicklungsumgebung informiert.

2.1 Beschaffung und Installation der Software

Für die Bearbeitung der Übungen und Aufgaben steht Ihnen die AVR-Entwicklungsumgebung SiSy AVR zur Verfügung. Sollten Sie SiSy AVR bereits installiert haben, können Sie dieses Kapitel überspringen.

2.1.1 Voraussetzungen

Für die Installation benötigen Sie einen Freischaltcode (Lizenzangaben). Falls Sie diese Angaben nicht mit der Software erhalten haben, können Sie diese online abrufen von

www.myAVR.de → Service

oder fordern Sie diese beim Hersteller an:

Tel: 03585-470222
Fax: 03585-470233
e-Mail: hotline@myAVR.de.

Außerdem sollten Sie prüfen, ob die Systemvoraussetzungen für die Installation und die Arbeit mit SiSy AVR gewährleistet sind.

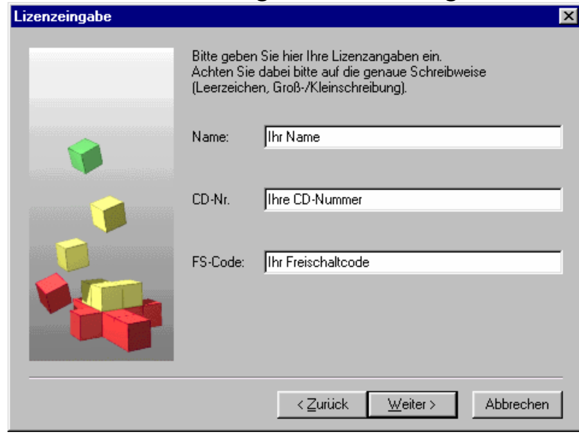
- für das Board 1.x einen PC-Arbeitsplatz oder Notebook mit LPT-Port und mindestens einem COM-Port
- für das Board 2.x einen PC-Arbeitsplatz oder Notebook mit USB-Anschluss
- ab Pentium III oder äquivalent
- 350 MB freier Speicherplatz auf der Festplatte
- Windows 2000, XP oder Vista (32-bit)
- 256 MB RAM
- Microsoft Internet-Explorer ab Version 6.0
- Maus oder ähnliches Zeigegerät
- Assembler Entwicklungsumgebung (in SiSy bereits integriert)
- myAVR Board
- LPT-Verlängerung (Board 1.x) bzw. USB Kabel (Board 2.x)
- Null-Modemkabel (bei Board 1.x)
- Bei Bedarf (z.B. autonomer Einsatz des myAVR Boards) geeignete Spannungsversorgung z.B. 9 V Batterie/stabilisiertes 9 V Netzteil

Des Weiteren sollten Sie Grundkenntnisse in einer beliebigen Programmiersprache besitzen. Die Installation und der erste Start müssen mit Administratorrechten durchgeführt werden.

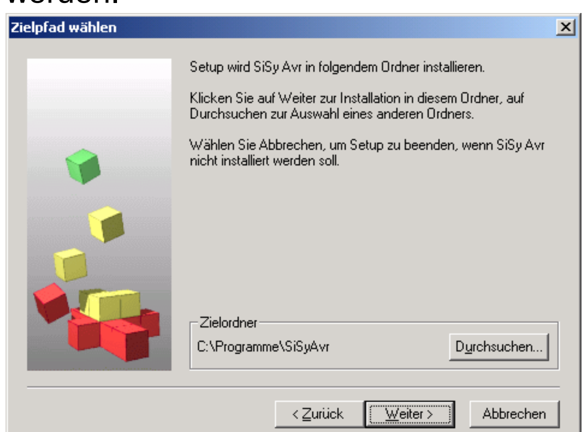
2.1.2 Setup von der SiSy-CD

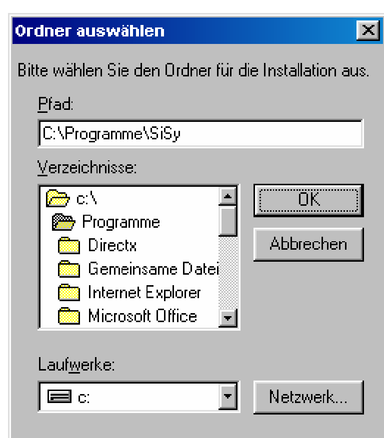
Legen Sie die CD „SiSy“ in Ihr CD-ROM-Laufwerk ein. Falls die CD nicht automatisch startet, wählen Sie bitte im Explorer das CD-ROM-Laufwerk und starten die exe-Datei aus der Wurzel des Laufwerks.
Auf dem Startbildschirm stehen Schaltflächen zur Verfügung zum Installieren der Software und zum Öffnen von Begleitdokumenten.
Für die Installation der Software betätigen Sie die entsprechende Schaltfläche. In Abhängigkeit Ihrer Rechnerkonfiguration kann der Start des Setup-Programms einige Sekunden dauern. Das gestartete Setup-Programm wird Sie durch die weitere Installation führen.

Beginn der Installation
Betätigen Sie im Setup-Programm die Schaltfläche „Weiter“. Sie erhalten die Lizenzbestimmungen. Bitte lesen Sie diese sorgfältig durch. Wenn Sie sich mit diesen Bestimmungen einverstanden erklären, bestätigen Sie die Lizenzbestimmungen.
Sie werden im folgenden Dialog dazu aufgefordert, Ihre Lizenzangaben einzugeben.



Danach erscheint folgende Dialogbox. Im unteren Teil wird der Pfad verwaltet, in dem SiSy zu installieren ist. Wenn ein anderer Pfad (bzw. ein anderes Laufwerk) gewünscht wird, ist die Schaltfläche „Durchsuchen“ zu betätigen. Eine Dialogbox erscheint; in den Feldern „Laufwerke“ und „Verzeichnisse“ können Laufwerk und Verzeichnis festgelegt werden.





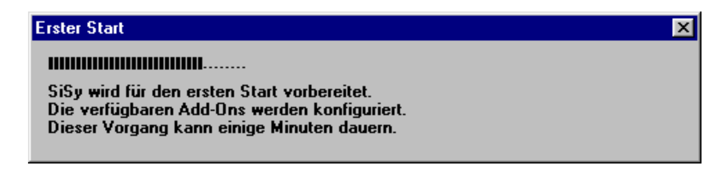
Wählen Sie danach den Programmordner, in dem die Verknüpfungen von SiSy eingefügt werden. Sie können den Zielordner ändern.

Setzen Sie die Installation mit „Weiter“ fort. Der für die Installation erforderliche Speicherplatz wird angezeigt und mit dem auf dem Laufwerk freien Speicherplatz verglichen. Wenn der freie Speicherplatz nicht ausreichend ist, erfolgt eine Warnung und Sie sollten den erforderlichen Speicherplatz zur Verfügung stellen.

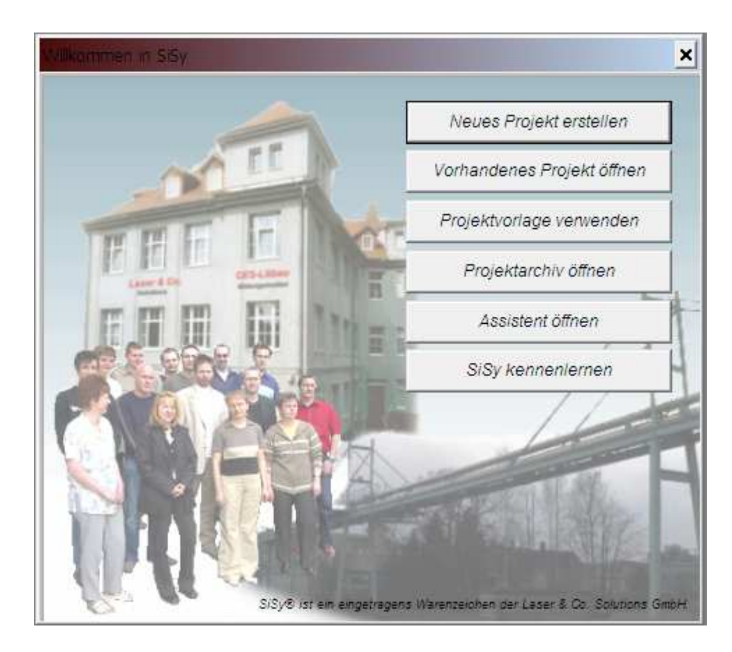
Hinweis:
In SiSy sind 2 Dateien enthalten, die Makros beinhalten („handbuch.doc“, „multi.doc"). Von einigen Virenscannern werden diese Makros als „Virus“ erkannt und entsprechend behandelt. In den Heuristik-Einstellungen des Virenscanners kann diese Behandlung unterdrückt werden.

Abschluss der Installation

Die Installation ist erst nach dem ersten Start von SiSy abgeschlossen. Dieser sollte im Anschluss an das Setup ausgeführt werden. Der Vorgang kann einige Minuten dauern.



Mit Beendigung dieses Vorgangs erscheint auf Ihren Bildschirm der Dialog „Willkommen in SiSy“. Folgen Sie dann den Hinweisen des Assistenten.



2.2 Beschaffen bzw. Herstellen der Hardware

Alle Ausführungen, Übungen und Aufgabenstellungen beziehen sich auf das myAVR Board als Referenzhardware. Wenn Sie Spaß an Elektronik haben, können Sie die Hardware auch selbst fertigen. Die Komponenten erhalten Sie als Bausatz oder fertig bestückt inklusive Schaltplan etc. unter www.myAVR.de.

Ausführliche Beschreibungen zur Programmierung mit Assembler, C/C++, BASCOM sind nicht Inhalt dieses Handbuches. Weiterführende Erklärungen dazu sind im „myAVR Lehrbuch Mikrocontrollerprogrammierung“ enthalten.

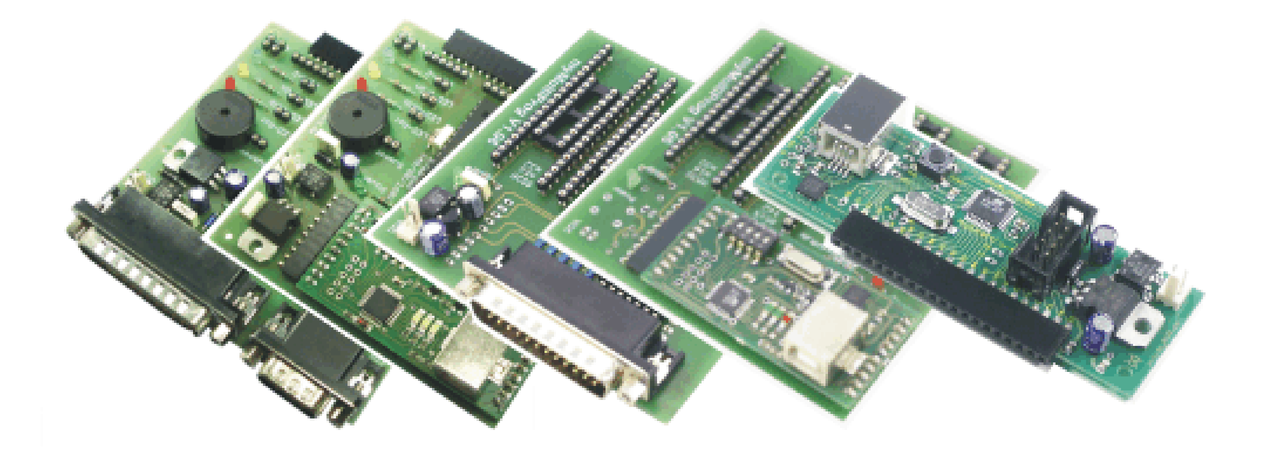


Abbildung 1: myAVR Board mit LPT- bzw. USB-Port sowie das myMultiProg und der mySmartControl

3 Arbeiten mit SiSy AVR, allgemein

Um dem Nutzer zu erläutern, wie er in SiSy modellieren kann, werden zweckentsprechende Definitionen der Begriffe gegeben.

3.1 Projektarbeit

3.1.1 Was ist ein SiSy-Projekt?

Ein SiSy-Projekt ist eine abgegrenzte Menge von verknüpften Elementen für ein zu bearbeitendes Problem. Alle aktuellen Daten sind in einer Projektdatenbank gespeichert. Die Projektdatenbank besteht aus einer Anzahl von Dateien im Projektverzeichnis, wobei jedes Projekt sein eigenes Verzeichnis hat. Durch das Anlegen eines Projektarchivs können diese in einer Datei komprimiert werden.

3.1.2 Neues Projekt erstellen

Ein neues Projekt wird eingerichtet und für die Bearbeitung bereitgestellt. Die Definition (Erstellung) eines neuen Projektes erfolgt durch Vergabe eines Projektdateinamens und/oder durch Überschreiben eines alten Projektes gleichen Namens. Über die Schaltfläche „Verzeichnis“ kann die vorgeschlagene Pfadangabe geändert werden.

Hinweis:
Falls ein bereits vorhandener Projekt- oder Verzeichnisname gewählt wird, erfolgt eine Überschreibwarnung.

Nach der Vergabe eines Projektnamens kann im nachfolgenden Fenster ein Vorgehensmodell ausgewählt werden. Je nach Vorgehensmodell können Vorlagen oder Assistenten das Erstellen eines neuen Projektes unterstützen.

3.1.3 Vorhandenes Projekt öffnen

Ein vorhandenes Projekt wird geöffnet und die abgespeicherten Diagramme und Objekte sowie alle Möglichkeiten für die weitere Bearbeitung werden verfügbar gemacht. Die Wahl des Projektes erfolgt durch Klicken auf den entsprechenden Namen oder über die Schaltfläche „Suchen“.

Beim Aktivieren der Checkbox „Beschreibung anzeigen“ werden zum markierten Projekt Informationen angezeigt. Dazu gehören außer der Beschreibung auch Informationen zum Bearbeitungsstand.

3.1.4 Projekt archivieren

Menü *Projekt/Archiv/Anlegen.*
Es kann ein komprimiertes Archiv des Projektes erzeugt werden. Dies ist besonders aus Gründen der Datensicherheit sinnvoll. Zielverzeichnis und Dateiname für die Archivdatei werden vorgeschlagen und können korrigiert werden. Der Umfang der Archivierung ist festzulegen und die Entscheidung für eine Komprimierung zu treffen, wobei diese empfohlen wird. Wenn ein Projekt unter einem bereits vorhandenen Archivnamen angelegt werden soll, wird eine Überschreibwarnung angezeigt. Bei Auswahl von „Nein“ wird die Erstellung des Archivs abgebrochen, bei „Ja“ wird das Projekt archiviert.

Hinweis: SiSy bietet die Möglichkeit des regelmäßigen Abspeicherns verschiedener Arbeitsstände, d.h. ein archiviertes Projekt wird nicht überschrieben. Ein Projektstand kann in einer neuen Archivdatei abgelegt werden

3.1.5 Projektarchiv einlesen

Menü Projekt/Archiv/Einlesen.
Hierunter versteht man das Einlesen eines Archivs zum Zweck der Rekonstruktion des Projektes.
Einlesen bedeutet Entpacken eines archivierten Projektes. Dazu sind der Archivpfad und der Dateiname des Archivs sowie das Zielverzeichnis anzugeben.
Hinweis:
Wenn im Zielpfad des Entpackens bereits ein Projekt existiert, erscheint eine Überschreibwarnung.

3.2 Die Modellierungselemente von SiSy

Werkzeug

SiSy stellt für die Bearbeitung der Modelle und Teilmodelle Werkzeuge der entsprechenden Methodik bereit. Werkzeuge sind Editoren, mit denen in einem Fenster die grafische Darstellung (Diagramme) der Modelle bearbeitet werden kann.

Diagramme

Diagramme sind grafische Darstellungen von Modellen oder Teilmodellen, die mit einem bestimmten Werkzeug erstellt werden. Die Modellierungselemente werden als Objekte in den Diagrammen unter Einhaltung von Regeln zusammengestellt.

Objekte

Objekte sind mögliche Modellelemente in Diagrammen, z.B. „kleines Programm“ in der „Programmierung“. Objekttypen sind konkrete Ausprägungen von Objekten, die in einem Diagramm angelegt wurden, z.B. Objekttyp „Lauflicht“ vom Objekt „kleines Programm“.

Referenzen

Die Objekte eines Diagramms können in anderen Diagrammen wiederverwendet werden. Durch das Hineinziehen aus dem Navigator oder aus einem offenen Diagramm wird eine Referenz vom Originalobjekt erzeugt. Die Referenz ist nur ein Verweis auf das Original, alle angezeigten Informationen wie Kurzname, Langname und Beschreibung werden vom Original bezogen. Somit sind die Informationen in allen Referenzen eines Objektes identisch mit dem Original. Dadurch werden Änderungen dieser Informationen automatisch auf alle Referenzen übertragen. Weiterhin ist es möglich, diese Referenzierung über einen sogenannten Report auszuwerten.

Kanten

Kanten sind Verbindungselemente zwischen Objekten. Eine Verbindung wird durch Ziehen mit der Maus (linke Maustaste) vom Verteiler des selektierten Objektes auf das gewünschte Objekt erreicht. Nach Loslassen der Maustaste und Prüfung der Verbindungszulässigkeit durch SiSy erscheint ein Kanten-Dialog, in dem das Element definiert und individuelle Einstellungen getroffen werden können.
Hinweis:
Bei Verbindung mit gehaltener STRG-Taste wird die Prüfung vernachlässigt und eine Zwangsverbindung erreicht.

Rahmen

Ein Rahmen fasst ausgewählte Objekte des Diagramms optisch zusammen. Er besitzt einen Kurz- sowie Langnamen und eine Objektbeschreibung, kann also als Objekt aufgefasst werden.
Hinweis:
Inhalte von Rahmen sind in Reports oder einer Dokumentengenerierung auswertbar.

3.3 Die Fenster für die Modellierung

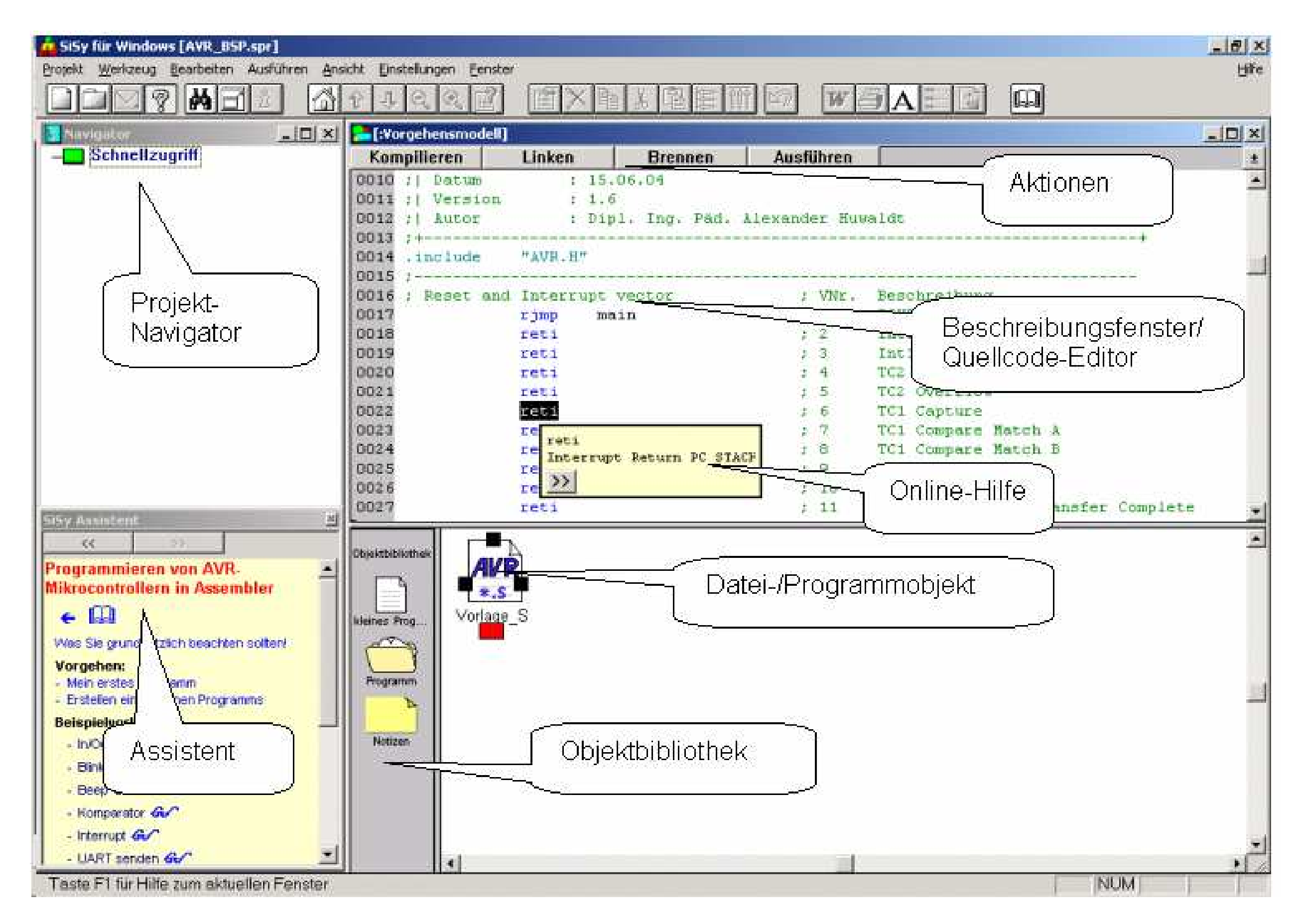


Abbildung 2

Navigator
Dieser befindet sich im linken, oberen Teil des Fensters. Er ermöglicht dem Anwender die Orientierung über die in der Projektdatenbank gespeicherten Objekte sowie deren Bereitstellung für die weitere Verwendung. Nach dem Start von SiSy werden neben dem Vorgehensmodell eine Reihe leicht zu handhabender Schaltflächen, Menüs und weitere Hilfsmittel angezeigt.

SiSy-Assistent
Am linken unteren Bildschirmrand befindet sich diese Nutzerunterstützung.

- Er begleitet den Anwender durch das gesamte Projekt und hält immer passende Informationen zur aktuellen Sicht parat
- Er ist auf die jeweilige Ausgabe von SiSy bezogen.
- Oft können Beispiele als Vorlagen vom Assistenten geladen werden.

Diagrammfenster
Am oberen Bildrand befinden sich das Hauptmenü, das hilfreiche Funktionen zu SiSy bereithält, und eine Werkzeugleiste, mit deren Hilfe schnell auf nützliche Anwendungen zugegriffen werden kann. Das Diagrammfenster nimmt die rechte Bildschirmseite ein und ist der Raum, in dem der Nutzer modelliert. Es enthält:

- das ausgewählte Vorgehensmodell,
- die Objektbibliothek mit den möglichen Objekten des aktuellen Diagramms sowie
- ein Fenster zur Beschreibung des markierten Objekts, in diesem Fall zum Editieren des Quelltextes.

Die Bedienelemente/Objektbibliothek

SiSy bietet, wie bei Windows-Anwendungen üblich, die Steuerung von Befehlen über das Hauptmenü, über die Werkzeugleisten, die Tastatur oder die Objektbibliothek an. Darüber hinaus enthalten das Kontextmenü und der Navigator Steuerfunktionen.

Die Anzahl der möglichen Befehle in der Menüleiste ist abhängig davon, ob ein Projekt geöffnet ist. Ist das nicht der Fall, erscheint ein Menü mit wenigen Befehlen. Bei einem geöffneten Projekt hält SiSy umfangreichere Menüs bereit. Die wichtigsten Menübefehle befinden sich auch als grafische Schaltfläche in der Werkzeugleiste, die eine schnelle und effiziente Bedienung des Programms ermöglicht. Die Toolbox-Darstellung bietet dem Anwender wichtige Programmfunktionen als direkten Link an.
Ein weiteres Bedienelement ist die Objektbibliothek. Sie unterstützt das Anlegen neuer Objekte.

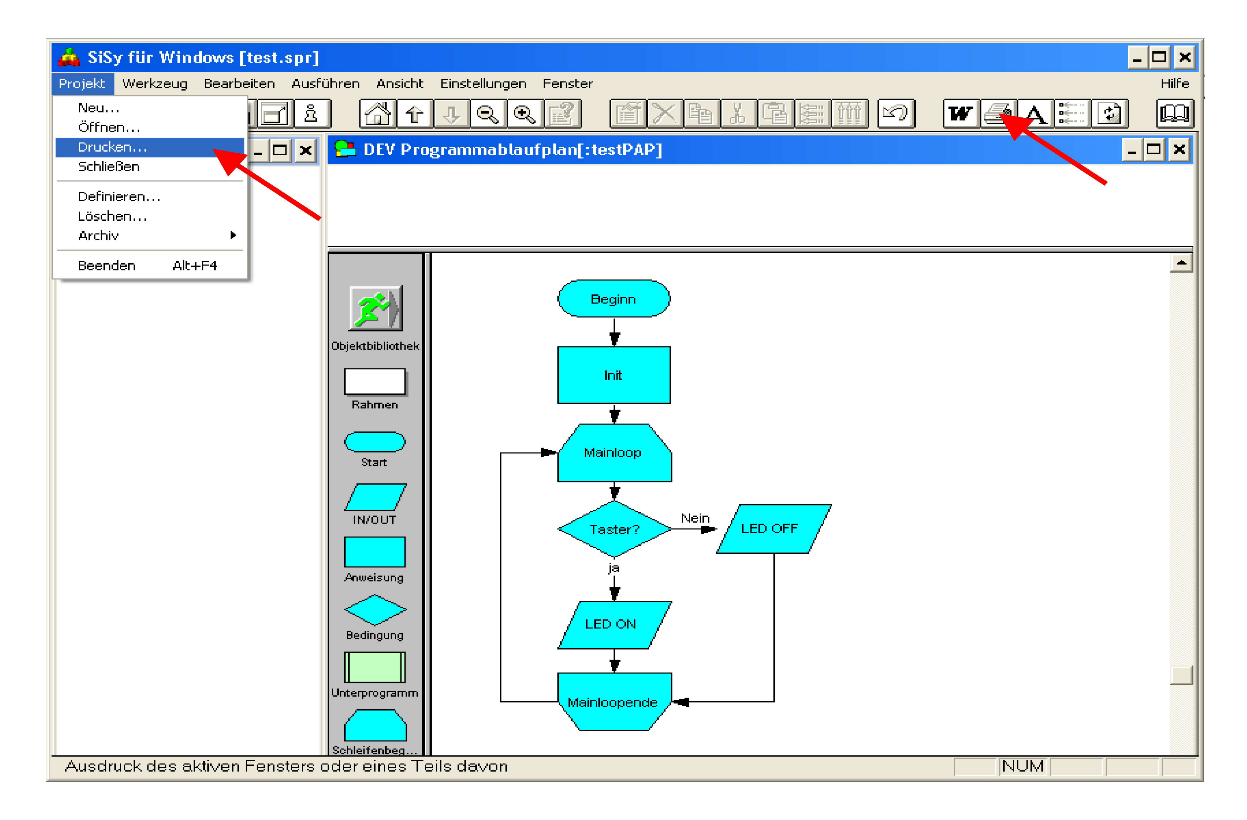
3.4 Druckfunktionen in SiSy

Sie haben in SiSy verschieden Möglichkeiten Projektinhalte wie Grafiken, Übersichten, Quellcodes oder ganze Projektdokumentationen zu drucken. Dabei ist zu berücksichtigen, dass in SiSy bestimmte Informationen wie zum Beispiel die Darstellung eines Programmablaufplanes sichtbare Elemente eines Diagramms sind und andere Teile wie zum Beispiel der Quellcode eines Elementes nur über Dialoge oder bei Selektierung des Elementes sichtbar sind. Je nachdem, welchen Inhalt Sie dokumentieren wollen, richtet sich die Auswahl der betreffenden Druckfunktion.

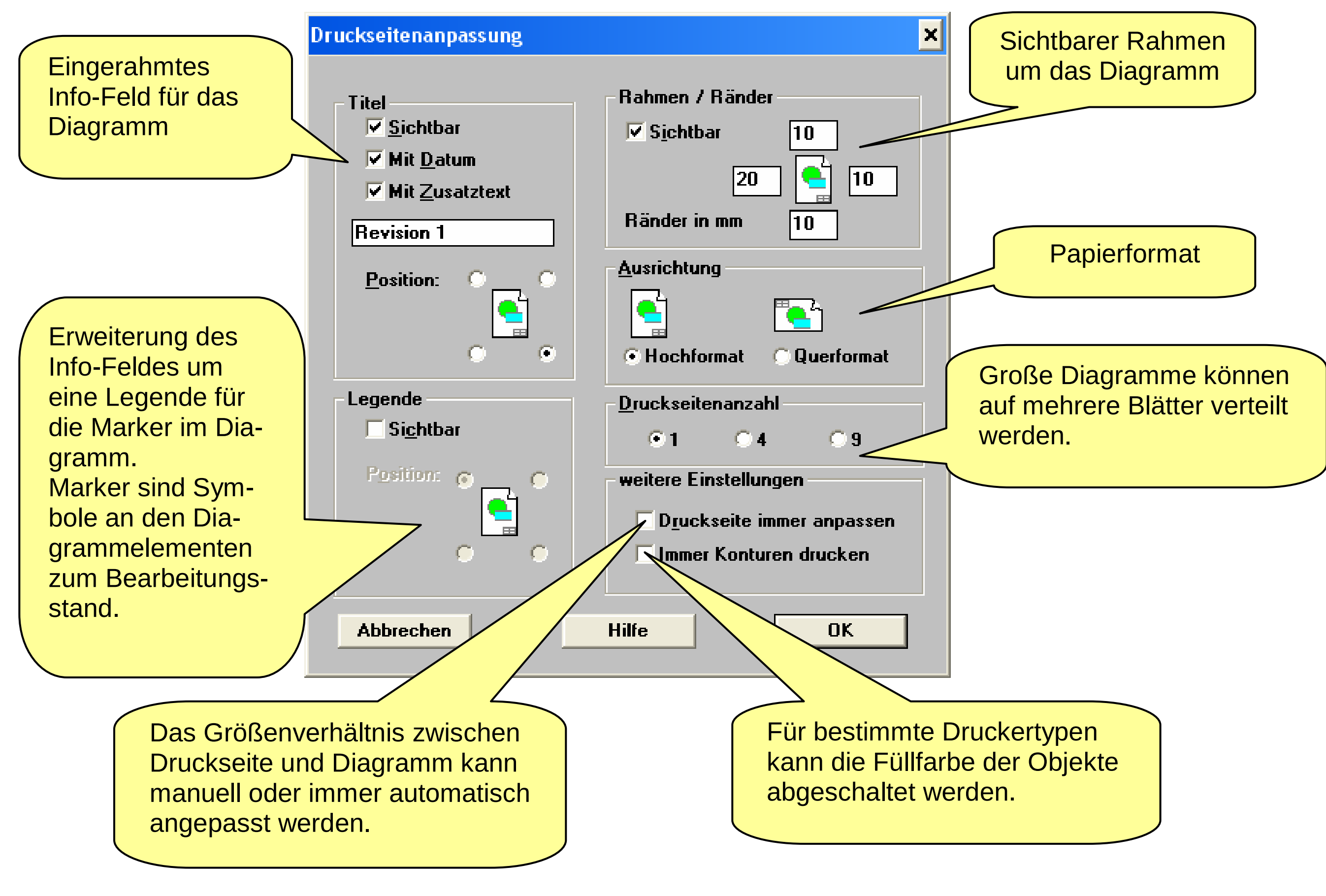
3.4.1 Diagramme drucken

Wenn Sie ein einzelnes Diagramm, also den sichtbaren Inhalt eines Diagrammfensters drucken wollen, gehen Sie wie folgt vor:

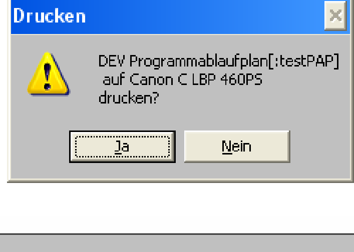
- ggf. Projekt öffnen
- das gewünschte Diagramm öffnen
- die Menüfolge *Projekt/Drucken* wählen oder das Druckersymbol in der Werkzeugleiste aktivieren



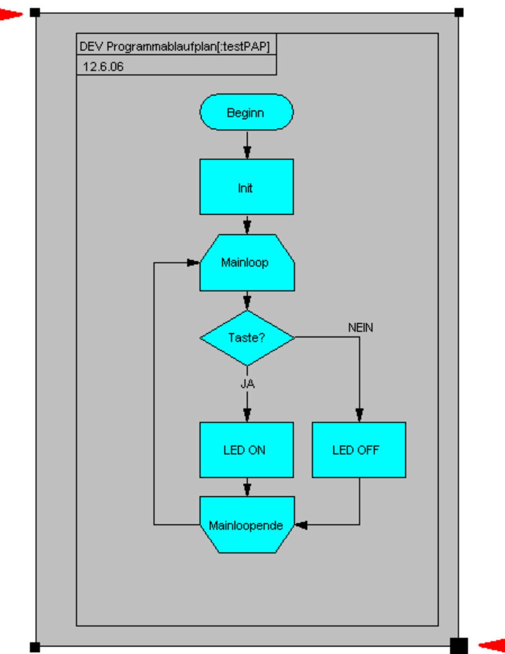
- Sie erhalten den Dialog zum Einrichten der Druckseite, wählen Sie die gewünschten Optionen aus!



- Zum Drucken wählen Sie die Schaltfläche OK. Der Ausdruck erfolgt auf dem Standarddrucker des Systems



- Im Hintergrund ist die Druckvorschau zu sehen. Der Druckvorgang kann hier abgebrochen werden, um die Einstellungen zu überarbeiten. Dabei kann die relative Position und das Größenverhältnis der Druckseite zum Diagramm verändert werden. Der Dialog zum Verändern der Einstellungen lässt sich per Doppelklick auf den Selektierungsmarken der Druckseite öffnen.



- Die Druckseitenansicht lässt sich über die Menüfolge *Ansicht/Druckseite* ein- und ausblenden

- Der Druckvorgang kann über das Druckersymbol in der Werkzeugleiste jederzeit gestartet werden.

3.4.2 Grafiken und Inhalte drucken (QuickDok)

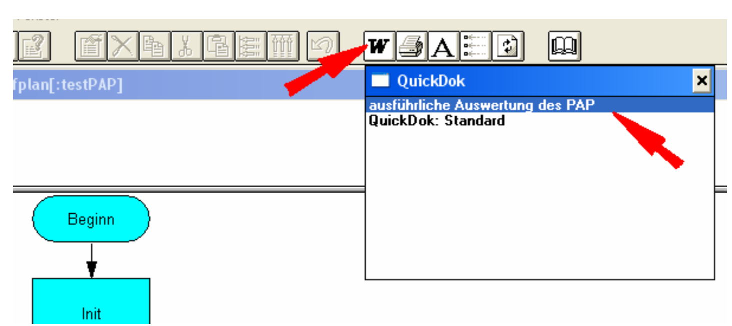
Viele Projektinformationen sind kein sichtbarer Teil von erstellten Diagrammen. Diese wurden über Dialoge und Masken eingegeben und stehen als Attribute in der Projektdatenbank zur Verfügung. Um diese Informationen auszudrucken, stellt SiSy für jeden Diagrammtyp eine Reportfunktion zur Verfügung, mit der die wesentlichen Informationen, Inhalte und Attribute des Diagramms und der Objekte in einem Diagramm, als Word-Dokument generiert werden.

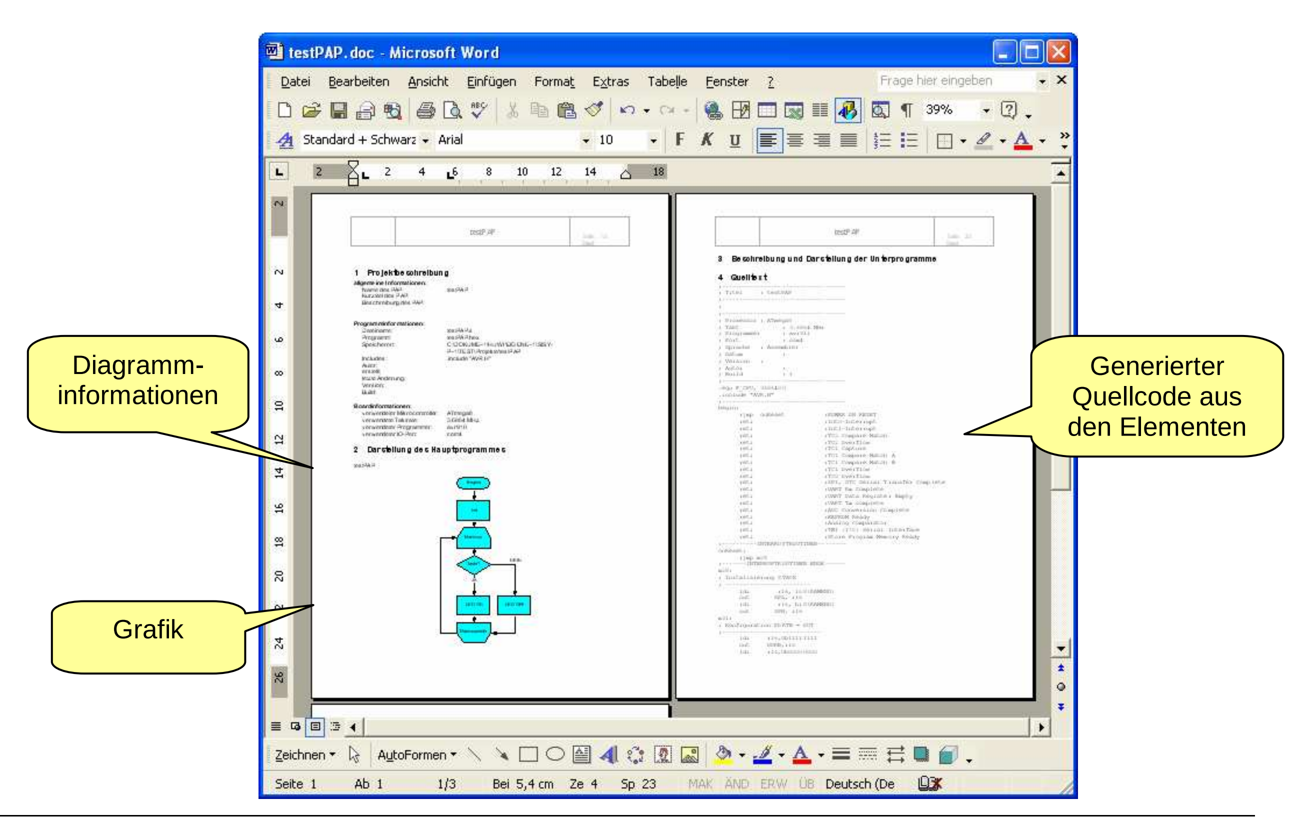
Für das Generieren des Word-Dokumentes muß auf dem PC MS Word installiert sein. Unterstützt werden die Versionen Word 2000/XP/2003.
Die Dokumentengenerierung basiert auf Makros; deshalb ist in Word der Makrovirenschutz auf *mittel* zu stellen.
Vorgehensweise in Word:

- Wählen Sie die Menüfolge *Extras/Makro/Sicherheit*
- Aktivieren Sie auf der Registerkarte „Sicherheitsstufe" *mittel*
- Auf der Registerkarte „Vertrauenswürdige Quellen" setzen Sie die Häkchen bei „Allen Vorlagen und Add-Ins vertrauen" und wenn vorhanden bei „Zugriff auf Visual Basic Projekte vertrauen"

Die Reportfunktion zum Generieren des Word-Dokumentes aktivieren Sie über das Symbol „QuickDok" in der Werkzeugleiste.

- Symbol (QuickDok) in der Werkzeugleiste wählen
- Aus der Liste der angebotenen Reportfunktionen auswählen
- Dateinamen für den Report vergeben
- Makros aktivieren

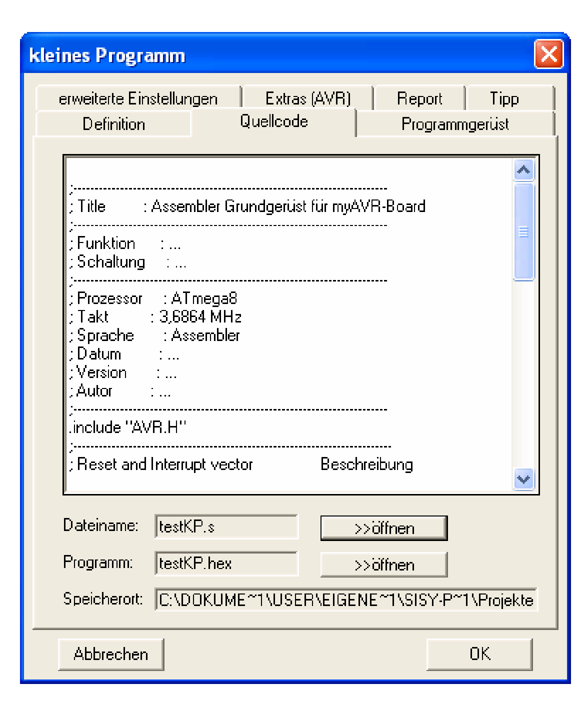




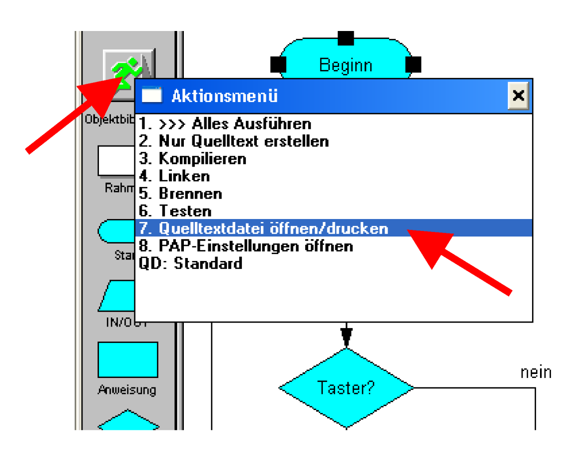
3.4.3 Nur Quellcodes drucken

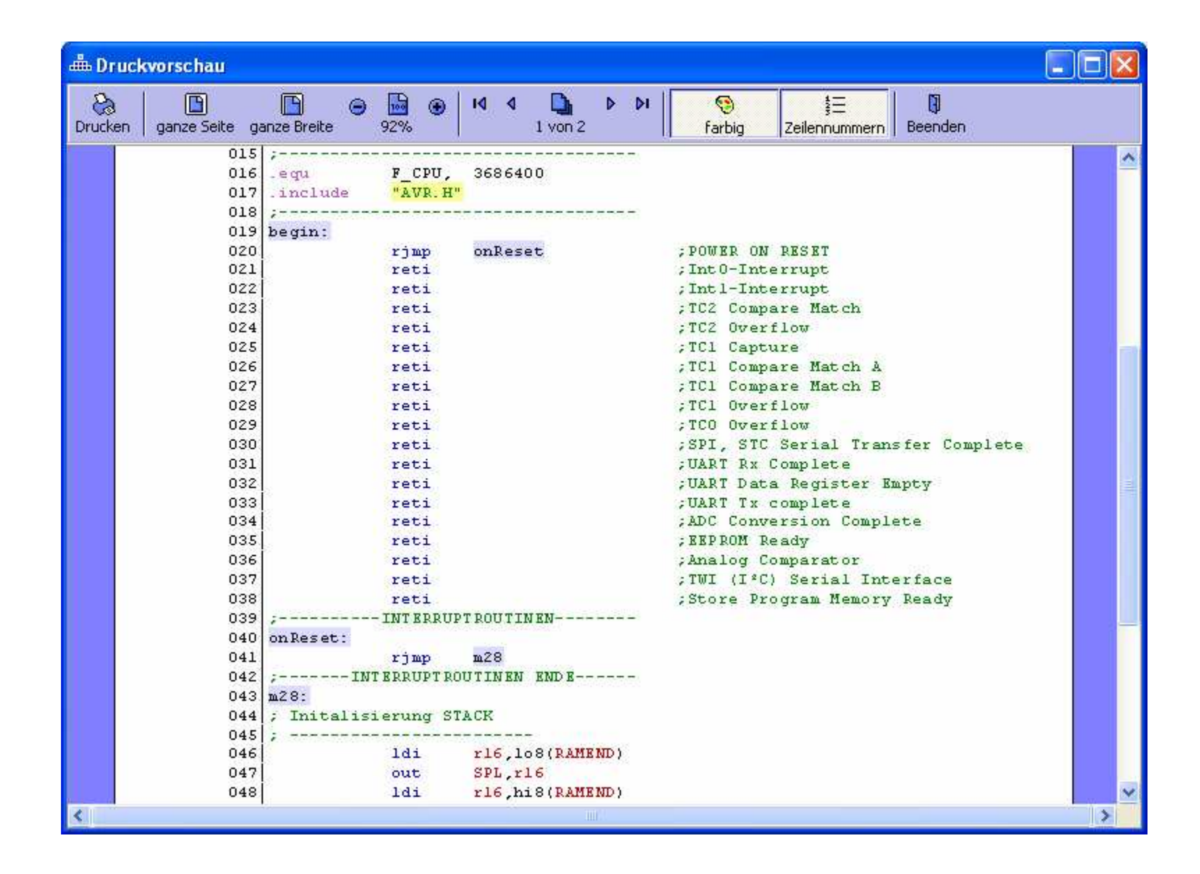
Für das Ausdrucken von Quellcodes bietet SiSy einen speziellen Quellcode-Druckassistenten. Damit ist es möglich Quellcodes formatiert, in Syntaxfarben und mit Zeilennummern auszudrucken.

- kleines Programm, Quellcode drucken
  - kleines Programm selektieren
  - rechte Maustaste „Definieren…“
  - Dialogseite „Quellcode“
  - Schaltfläche „>>öffnen“



- Programmablaufplan, Quellcode drucken
  - PAP öffnen
  - Ampelmänchen aktivieren
  - „Quellcodedatei öffnen/drucken“

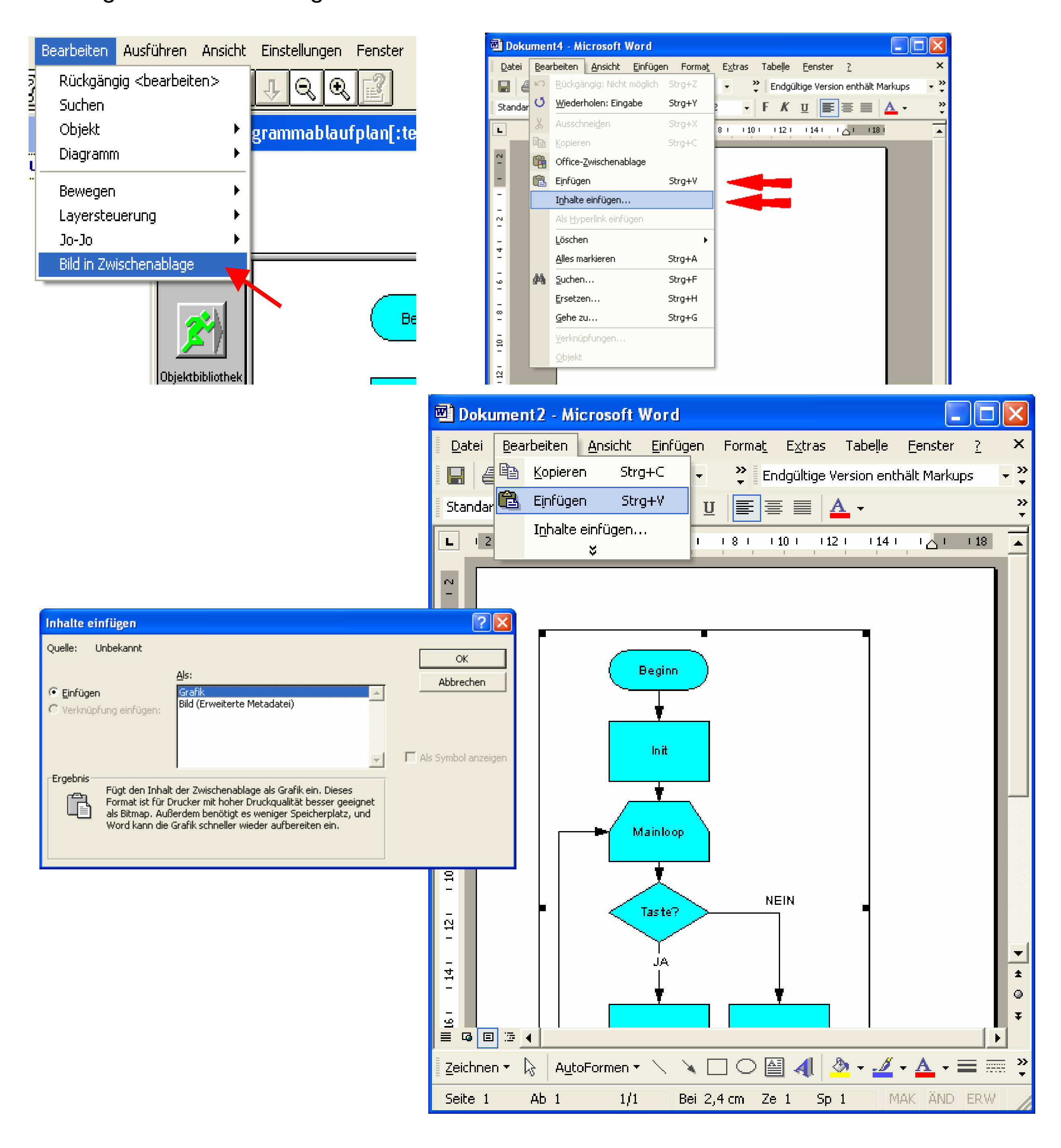




3.4.4 Nutzen der Zwischenablage

Oft ist es erforderlich in Projektdokumentationen die Diagramme als Bilder einzufügen. In SiSy werden die Diagramme nicht als Bilder gespeichert sondern zur Laufzeit aus den Modellinformationen generiert. Um die Bilder der Diagramme weiter zu verwenden, steht dem Anwender die Funktion „Bild in Zwischenablage“ zur Verfügung. Dabei erstellt SiSy eine skalierbare Vektorgrafik (WMF) und legt diese in die Zwischenablage (Copy). Die Grafik kann nun von anderen Anwendungen über den Befehl „Einfügen“ (Past) beliebig weiter verwendet werden.

- gewünschtes Diagramm öffnen
- Menüfolge *Bearbeiten/Bild in Zwischenablage* wählen
- Zielanwendung, zum Beispiel Word öffnen
- Menüfolge *Bearbeiten/Einfügen* oder *Bearbeiten/Inhalte einfügen* wählen
- Gegebenfalls einzufügendes Grafikformat wählen



4 Die Hilfefunktionen in SiSy

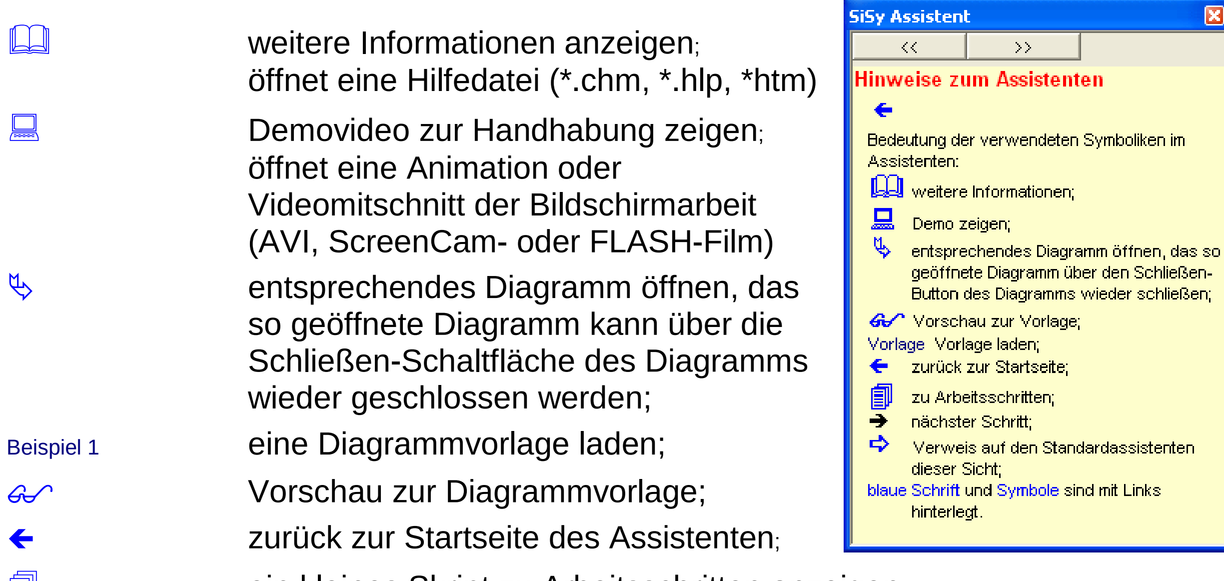
Nutzen Sie die zahlreichen Hilfen und Vorlagen, die SiSy dem Entwickler bietet!

4.1 Der Assistent

Der Assistent ist hilfreich bei der Unterstützung und Führung des Nutzers im Umgang mit SiSy. Er befindet sich standardmäßig im linken, unteren Bildschirmbereich. Der Assistent kann entweder über das Menü *Ansicht/Assistent* oder über die Werkzeugleiste geöffnet werden, falls dies nicht beim Programmstart erfolgte. Der Assistenten begleitet Sie im gesamten Projekt. Sie erhalten immer passende Informationen zum aktuellen Diagrammtyp und haben die Möglichkeit, durch Links weitere Hilfethemen aufzurufen oder Vorlagen in Ihr Projekt zu laden.

Beachte: Der Assistent ist auf die jeweilige Ausgabe von SiSy, die verfügbaren Add-Ons und das gewählte Modell bezogen.

Bedeutung der verwendeten Symboliken im Assistenten:



(Buch-Symbol)	weitere Informationen anzeigen; öffnet eine Hilfedatei (*.chm, *.hlp, *htm)
(Bildschirm-Symbol)	Demovideo zur Handhabung zeigen; öffnet eine Animation oder Videomitschnitt der Bildschirmarbeit (AVI, ScreenCam- oder FLASH-Film)
(Pfeil-Symbol)	entsprechendes Diagramm öffnen, das so geöffnete Diagramm kann über die Schließen-Schaltfläche des Diagramms wieder geschlossen werden;
Beispiel 1	eine Diagrammvorlage laden;
(Brillen-Symbol)	Vorschau zur Diagrammvorlage;
←	zurück zur Startseite des Assistenten;
(Skript-Symbol)	ein kleines Skript zu Arbeitsschritten anzeigen;
➔	nächster Schritt (Arbeitsschritt);
⇨	Verweis auf den Standardassistenten dieser Sicht;
blaue Schrift	und
blaue Symbole	sind mit Links hinterlegt und können per Mausklick aktiviert werden.

Beispiele für Assistenten:

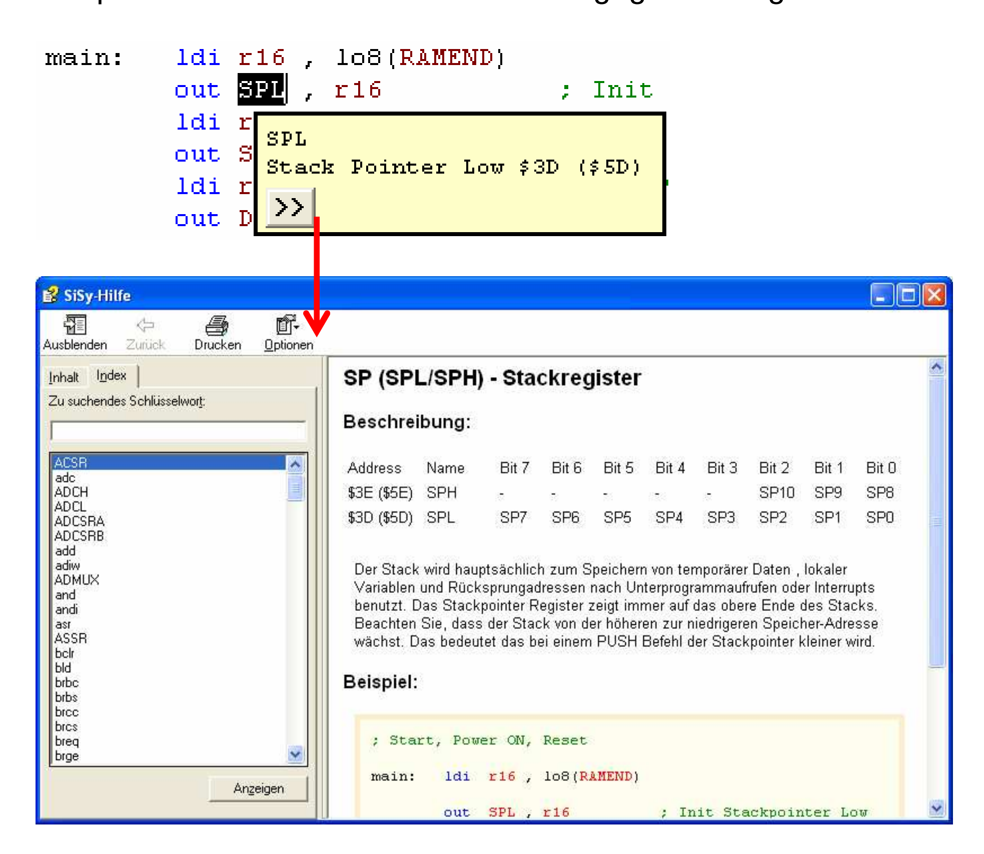


4.2 Die Online-Hilfe

Bei der Eingabe von Quellcodes im dafür vorgesehenen Editorfenster werden reservierte Worte (Bezeichner, Schlüsselworte) der gewählten Programmiersprache durch verschiedenfarbiges Einfärben (Syntaxfarben) hervorgehoben. Zu den hervorgehobenen Bezeichnern existiert in der Regel auch eine kurze Online-Hilfe und eine ausführlichere Hilfe. Die Online-Hilfe ist ein Pop-Up-Fenster, welches bei der Eingabe des Bezeichners automatisch eingeblendet wird. In dem Pop-Up ist eine kurze Hilfestellung zu dem Bezeichner eingeblendet. Die Online-Hilfe ist auch durch Doppelklick auf dem Bezeichner erreichbar. Steht eine Hilfe mit ausführlicheren Informationen zur Verfügung, wird in dem Pop-Up eine Schaltfläche [**>>**] zum Öffnen des Hilfethemas angeboten.

Register-Hilfe

Bei der Eingabe von bekannten Registernamen wird die Bezeichnung des Registers und dessen Adresse eingeblendet. Die Schaltfläche [**>>**] öffnet eine Hilfedatei mit dem entsprechenden Hilfethema zu dem eingegeben Register.



Bit-Hilfe

Wenn für ein Register detaillierte Informationen zu der Bedeutung/Funktion der einzelnen Bits vorliegen, wird zusätzlich im Pop-Up eine Kurzreferenz der Bits angezeigt. Die Schaltfläche **[>>]** öffnet eine Hilfedatei mit dem entsprechenden ausführlichen Hilfethema.

```
ACSR
 ACSR
 Analog Comparator Control and Status Register $08 ($28)
 7    6     5    4    3     2    1      0
 ACD  ACBG  ACO  ACI  ACIE  ACIC ACIS1  ACIS0
 >>
```

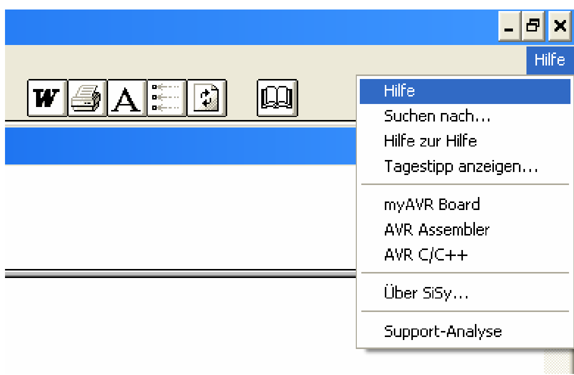
Befehlshilfe

Bei der Eingabe von Befehlen wird in der Regel die Bedeutung bzw. Funktion des Befehls und ein kurzes Syntaxbeispiel eingeblendet. Die Schaltfläche **[>>]** öffnet eine Hilfedatei mit dem entsprechenden ausführlichen Hilfethema.

```
brne    skip
brne  marke
Branch if Not Equal if (Z = 0) then PC PC + k + 1
>>
```

4.3 Die allgemeine Hilfe

SiSy bietet neben der direkten Hilfe bei der Eingabe von Schlüsselworten auch allgemeine Hilfen an.



In SiSy AVR sind das zum Beispiel die allgemeine Hilfe zum Modellierungstool SiSy und die Hilfen zum myAVR Board, dem AVR Assembler und AVR C. Die Hilfedateien verfügen über einen Index sowie die Möglichkeit der Volltextsuche.

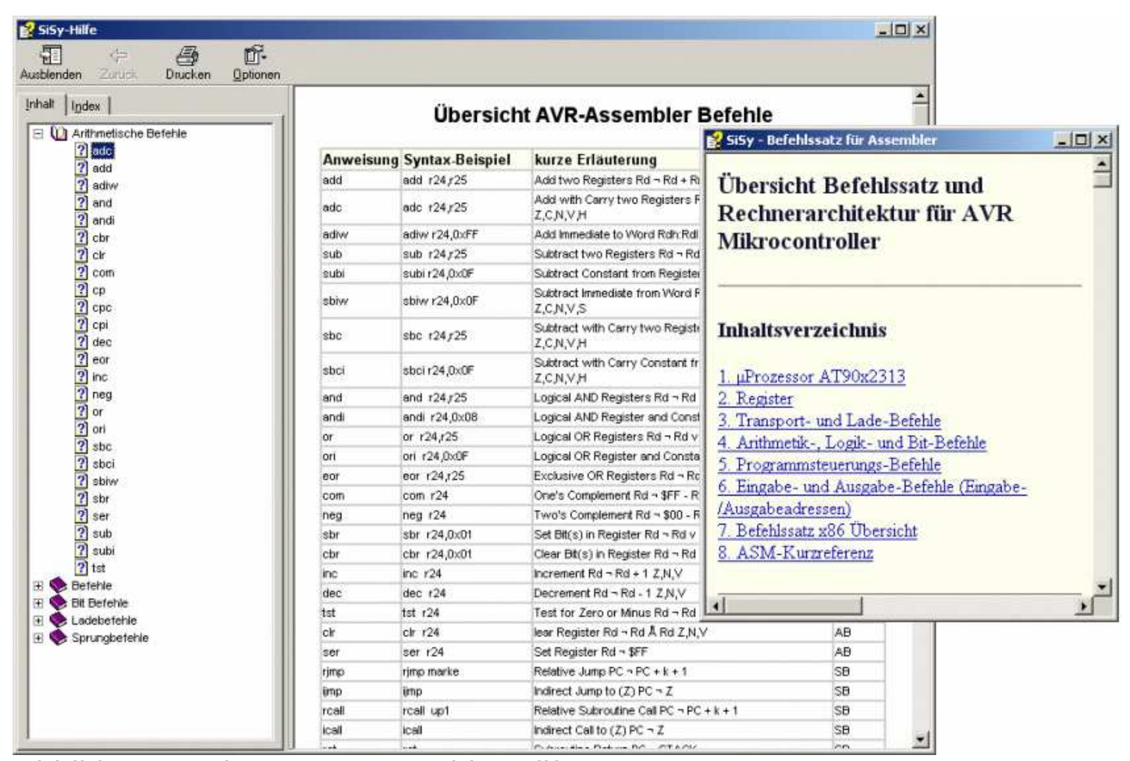


Abbildung 3: SiSy AVR-Assembler Hilfe

5 Entwicklung eines kleinen Programms mit SiSy AVR

Schauen wir uns als nächstes kurz in der Entwicklungsumgebung SiSy AVR um. Für die Entwicklung von Mikrocontrollerlösungen bietet sich für den Einstieg die einfache Programmierung (kleines Programm) an.

5.1 Voraussetzungen

Für die Bearbeitung der folgenden Aufgaben benötigen Sie die aufgeführte Software und Hardware.

Software:
- SiSy AVR ab Version 2.18 oder
- SiSy Ausgabe Developer, Professional oder BS und das installierte Add-On AVR

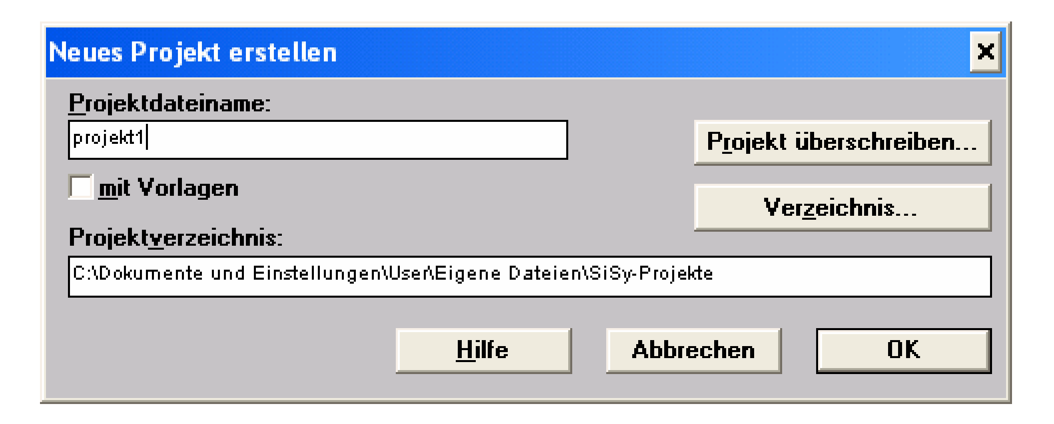
Hardware:
- Ein bestücktes myAVR Board
- Programmierkabel (USB bzw. LPT)
- eventuell Nullmodemkabel für die COM Schnittstelle
- 9 V Batterie bei Bedarf (z.B.: autonomer Einsatz)
- Patchkabel

5.2 Zielstellung

Der Schnelleinstieg zur Mikrocontroller-Programmierung soll Ihnen helfen, SiSy AVR kennen zu lernen und erste Schritte in der hardwarenahen Programmierung mit SiSy zu gehen.
Zielstellung für das erste Beispielprogramm: Die drei LED's auf dem myAVR Board sollen nacheinander aufleuchten und damit ein „Lauflicht" erzeugen.

5.3 Vorgehen

Ein neues Projekt anlegen
Starten Sie SiSy und wählen Sie *Assistent öffnen*. Danach klicken Sie im SiSy-Assistent auf den Menüpunkt *Neues Projekt anlegen*, vergeben Sie den Projektnamen „Lauflicht". Bestätigen Sie diesen Dialog mit *OK*. Wählen Sie als nächstes das Vorgehensmodell „Programmierung". Es folgt ein Fenster, welches Sie mit *Weiter* bestätigen. Als nächstes wählen Sie Ihren Programmer und Ihren Controller aus. Klicken Sie nun auf *Speichern*. Den folgenden Dialog bestätigen Sie mit *OK.* Im darauf folgenden Fenster wählen Sie die Taktrate Ihres Mikrocontrollers aus. Mit einem Klick auf *Fertig stellen* beenden Sie dieses Fenster. Daraufhin erscheint ein Auswahlfenster. Wählen Sie *„leeres Diagramm"* aus. Beenden Sie das Fenster über *Weiter* sowie *Fertig stellen*.

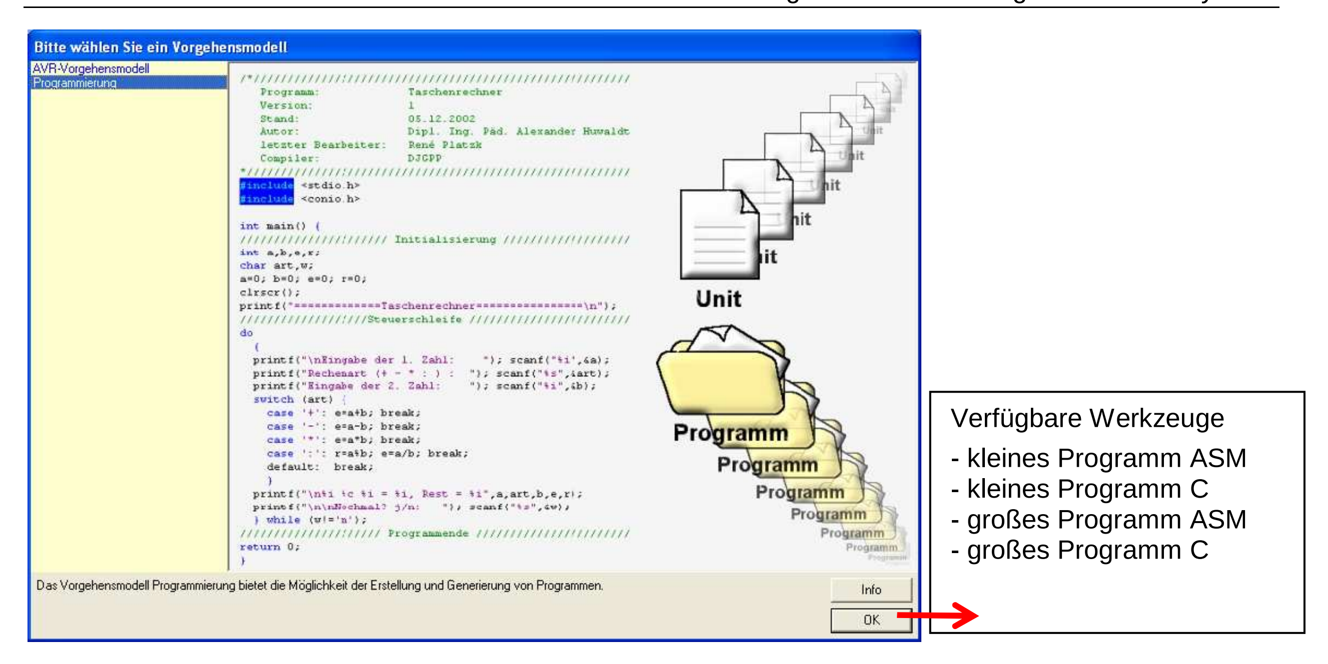


Hinweis:
In SiSy legen Sie stets ein Projekt an. In dieses Projekt integrieren Sie Ihr Programm bzw. mehrere Programme. Unabhängig vom Programmnamen benötigt jedes Projekt einen Namen.
Wenn in ihrer SiSy Ausgabe mehrere Vorgehensmodelle (Hauptebenen) zur Verfügung stehen, wird eine Auswahl der Hauptebene angeboten. Mit der Auswahl des Vorgehensmodells (VGM) entscheiden Sie über das Profil des Projektes. Wenn ein Projekt klein und die Modellierung von Programmablaufplänen nicht erforderlich ist, wählt man das einfachere Vorgehensmodell; in dem gezeigten Beispiel „Programmierung“. Bei komplexen Projekten, für die neben kleinen Programmen auch Programmablaufpläne, Struktogramme oder Klassendiagramme nötig sind, wird das komplexere Vorgehensmodell „AVR-Vorgehensmodell“ ausgewählt.
Die Auswahl des Vorgehensmodells bestimmt im weiteren Projektverlauf die von SiSy zur Verfügung gestellten Werkzeuge. Bei Auswahl des einfachen Vorgehensmodells „Programmierung“ stehen dann zum Beispiel keine grafischen Werkzeuge wie der Programmablaufplan zur Verfügung. Damit sind die Menüs und Objektbibliotheken entsprechend übersichtlicher.

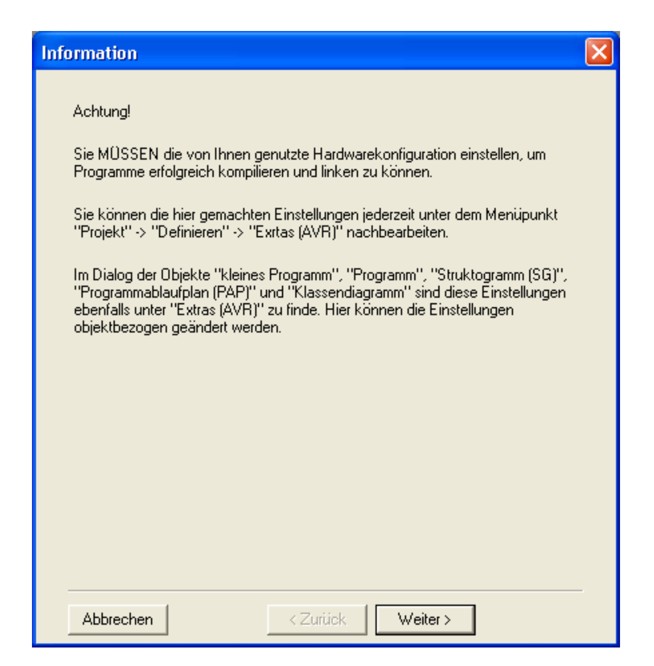


Verfügbare Werkzeuge

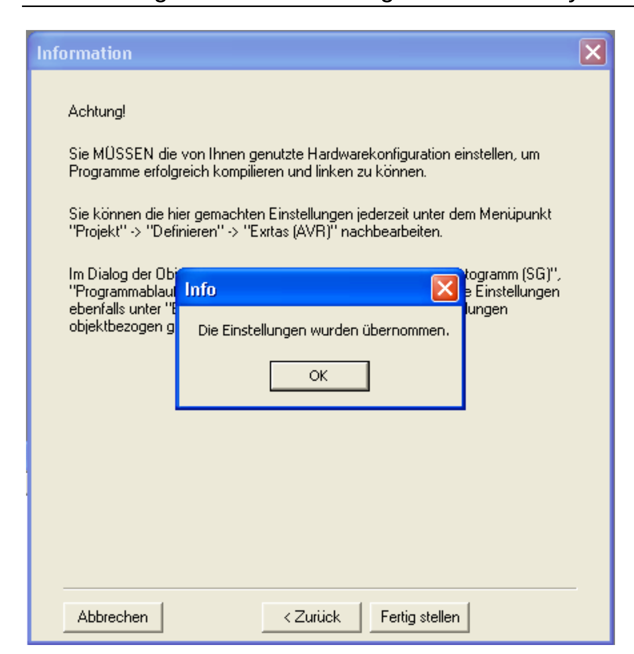
- kleines Programm ASM
- kleines Programm C
- großes Programm ASM
- großes Programm C
- Programmablaufplan ASM
- Struktogramm C
- Klassendiagramm C++

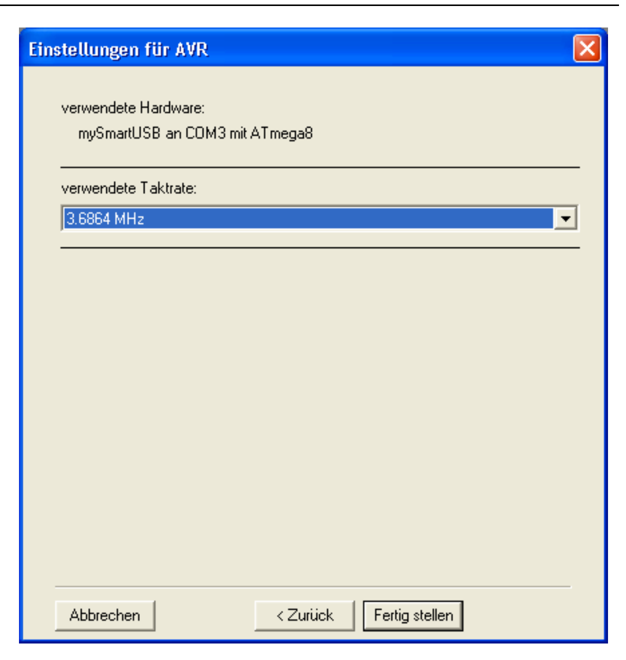


Im weiteren Verlauf werden entsprechend der Auswahl des Vorgehensmodells unterschiedliche Hilfen zum Erstellen eines Programms angeboten. Für Einsteiger empfiehlt sich zuerst die Nutzung des Vorgehensmodells „Programmierung“.
Bei jedem neuen Projekt müssen die Grundeinstellungen zur verwendeten Zielplattform vorgenommen werden (Mikrocontrollertyp, Taktrate, Programmer und IO-Port).









Werden keine Einstellungen vorgenommen, so legt SiSy folgende Standardwerte fest:

Board :	mySmartUSB
Port:	COM 3
Mikrocontroller:	ATmega8
Taktrate:	3,6864 MHz

Kleines Assembler-Programm anlegen

Erstellen Sie ein Programm für den AVR-Mikrocontroller, indem Sie per Drag & Drop aus der Objektbibliothek ein „kleines Programm“ in das Diagrammfenster ziehen. Das Kontextmenü öffnet sich nun automatisch. Für spätere Bearbeitungen, markieren Sie das Objekt und wählen Sie aus dem Kontextmenü (rechte Maustaste) *Definieren*.

Auf der Registerkarte „Definition“ tragen Sie den Programmnamen ein (im Beispiel „Lauflicht“) und wählen die Programmiersprache aus, hier „AVR Assembler“; siehe Abbildung 4.



Abbildung 4: Definition kleines Programm

Zielplattform und Programmer einstellen

Kontrollieren Sie auf der Registerkarte „Extras (AVR)“ den ausgewählten Mikrocontroller, siehe Abbildung 5. Die Option „Vorgaben benutzen“ überträgt automatisch die Grundeinstellungen (Mikrocontrollertyp, Taktrate, Programmer und IO-Port) des Projektes in die lokalen Einstellungen. Sollen die lokalen Einstellungen unter „Extras (AVR)“ von den Projekteinstellungen abweichen, muss die Option „Vorgaben benutzen“ abgeschaltet werden.

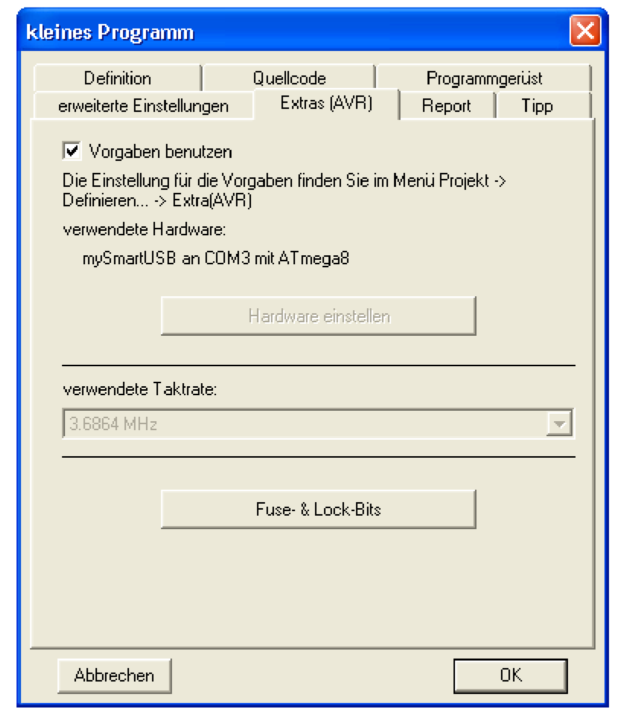


Abbildung 5: Einstellungen Extras (AVR)

Beachte folgende Einstellungen:

- myAVR-Board 1 LPT
  - Programmer: SP12
  - Port: LPTx (z.B. LPT1)
- myAVR-Board 2 USB / mySmartUSB
  - Programmer: AVR911
  - Port: COMx (z.B. COM3)
- mySmartUSB mit Firmware ab 1.5
  - Programmer: AVR911
  - Port: COMx (z.B. COM3)

Programmgerüst laden, Quellcode erstellen

Über die Registerkarte „Programmgerüst“ können Sie das folgende Grundgerüst für ein AVR Assemblerprogramm laden; in der Registerkarte „Quellcode“ können Sie den Quellcode eigenständig eintragen.

<u>*Hinweis:*</u>
Den Quellcode können Sie auch im Beschreibungsfenster/Editorfenster der SiSy-Benutzeroberfläche eintragen bzw. korrigieren.

Laden Sie die Vorlage für das Programmgerüst oder erstellen Sie den folgenden Quellcode.
Vergleichen Sie dazu auch den Abschnitt zum myAVR Code-Wizard (Abschnitt 7). Das Programmgerüst darf erst geladen werden, wenn der Zielcontroller ausgewählt wurde; die Vorlagen sind controllerspezifisch.

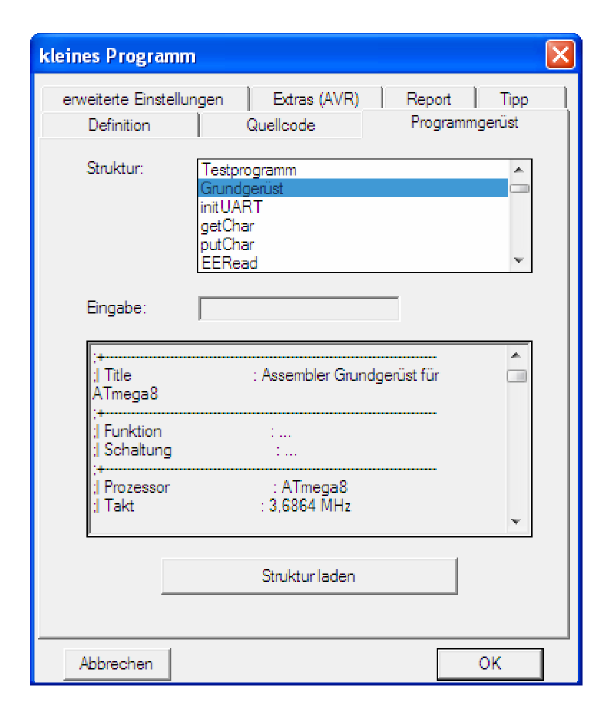


Abbildung 6: Grundgerüst laden

```
;+----------------------------------------------------------------------------
;| Title                           : Assembler Grundgerüst für ATmega8
;+----------------------------------------------------------------------------
;| Prozessor                       : ATmega8
;| Takt                            : 3,6864 MHz
;| Sprache                         : Assembler
;+----------------------------------------------------------------------------
.include        "AVR.H"
;-----------------------------------------------------------------------------
begin:      rjmp   main                ;  RESET External Pin, Power-on Reset,
                                          Brown-out Reset and Watchdog Reset
                   reti                ;  INT0 External Interrupt Request 0
                   reti                ;  INT1 External Interrupt Request 1
                   reti                ;  TIMER2 COMP Timer/Counter2 Compare Match
                   reti                ;  TIMER2 OVF Timer/Counter2 Overflow
                   reti                ;  TIMER1 CAPT Timer/Counter1 Capture Event
                   reti                ;  TIMER1 COMPA Timer/Counter1 Compare Match A
                   reti                ;  TIMER1 COMPB Timer/Counter1 Compare Match B
                   reti                ;  TIMER1 OVF Timer/Counter1 Overflow
                   reti                ;  TIMER0 OVF Timer/Counter0 Overflow
                   reti                ;  SPI, STC Serial Transfer Complete
                   reti                ;  USART, RXC USART, Rx Complete
                   reti                ;  USART, UDRE USART Data Register Empty
                   reti                ;  USART, TXC USART, Tx Complete
                   reti                ;  ADC ADC Conversion Complete
                   reti                ;  EE_RDY EEPROM Ready
                   reti                ;  ANA_COMP Analog Comparator
                   reti                ;  TWI 2-wire Serial Interface
                   reti                ;  SPM_RDY Store Program Memory Ready
;-----------------------------------------------------------------------------
main:              ldi          r16,hi8(RAMEND)      ; Main program start
                   out          ioSPH,r16            ; Set Stack Pointer to top of RAM
                   ldi          r16,lo8(RAMEND)
                   out          ioSPL,r16
                   ;Hier Init-Code eintragen.
;-----------------------------------------------------------------------------
mainloop:   wdr
                   Hier den Quellcode eintragen.
                   rjmp   mainloop
```

Quellcode in Assembler erstellen

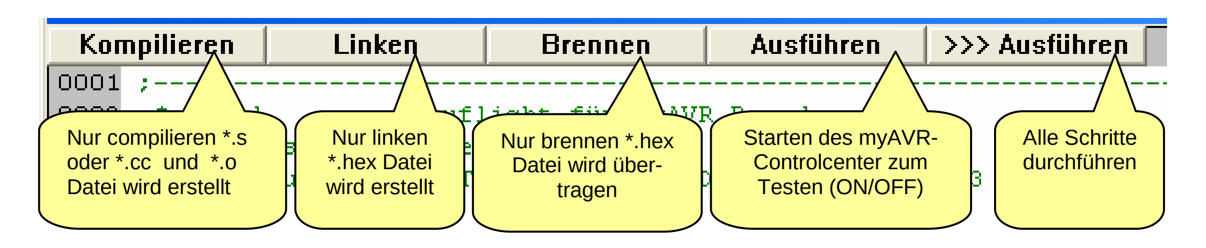
Das Lauflicht soll über die Ausgabegeräte LED angezeigt und von dem Prozessorport D gesteuert werden. Die Realisierung erfolgt über je ein Bit im Register R18. Dieses wird mit dem Befehl Bit-Rotation nach rechts verschoben und an PORT D des Prozessors ausgegeben. Auf Grund der Prozessorgeschwindigkeit muss die Ausgabe des Lauflichtes für unser Auge verzögert werden. Geben Sie folgenden Quellcode ein bzw. ergänzen Sie die Programmvorlage!

```
;---------------------------------------------------------------------------------
;* Titel            :Lauflicht für myAVR Board
;* Prozessor        :ATmega8 mit 3,6864 MHz
;* Schaltung        :PORT D.2 bis PORT D.4 an LED 1 bis 3
;* Datum            :15.03.2004
;* Autor            :Dipl. Ing. Päd. Alexander Huwaldt
;---------------------------------------------------------------------------------
.include            "avr.h"
;---------------------------------------------------------------------------------

; Reset and Interruptvectoren    ; VNr. Beschreibung
begin:       rjmp    main        ; 1    POWER ON RESET
             reti                ; 2    Int0-Interrupt
             reti                ; 3    Int1-Interrupt
             reti                ; 4    TC2 Compare Match
             reti                ; 5    TC2 Overflow
             reti                ; 6    TC1 Capture
             reti                ; 7    TC1 Compare Match A
             reti                ; 8    TC1 Compare Match B
             reti                ; 9    TC1 Overflow
             reti                ; 10   TC0 Overflow
             reti                ; 11   SPI, STC Serial Transfer Complete
             reti                ; 12   UART Rx Complete
             reti                ; 13   UART Data Register Empty
             reti                ; 14   UART Tx complete
             reti                ; 15   ADC Conversion Complete
             reti                ; 16   EEPROM Ready
             reti                ; 17   Analog Comparator
             reti                ; 18   TWI (I²C) Serial Interface
             reti                ; 19   Store Program Memory Redy
;---------------------------------------------------------------------------------
; Start, Power ON, Reset
main:        ldi    r16 , lo8(RAMEND)
             out    SPL , r16             ; Init Stackpointer LO
             ldi    r16 , hi8(RAMEND)
             out    SPH , r16             ; Init Stackpointer HI
             ldi    r16 , 0b11111111      ; PORT D auf Ausgang
             out    DDRD , r16
             ldi    r16 , 0b00000000      ; Alle Bits auf LOW
             out    PORTD , r16
             ldi    r17 , 0b00000000
             ldi    r18 , 0b00000001      ; 1 Lauflicht-Bit
;---------------------------------------------------------------------------------
mainloop:    wdr
             inc    r16                   ; Wait
             brne   skip
             inc    r17                   ; Wait
             brne   skip
             rcall  up1                   ; Lauflicht
skip:        rjmp   mainloop
;---------------------------------------------------------------------------------
up1:         rol    r18                   ; Bit-Rotation
             out    PORTD , r18
             ret
;---------------------------------------------------------------------------------
```

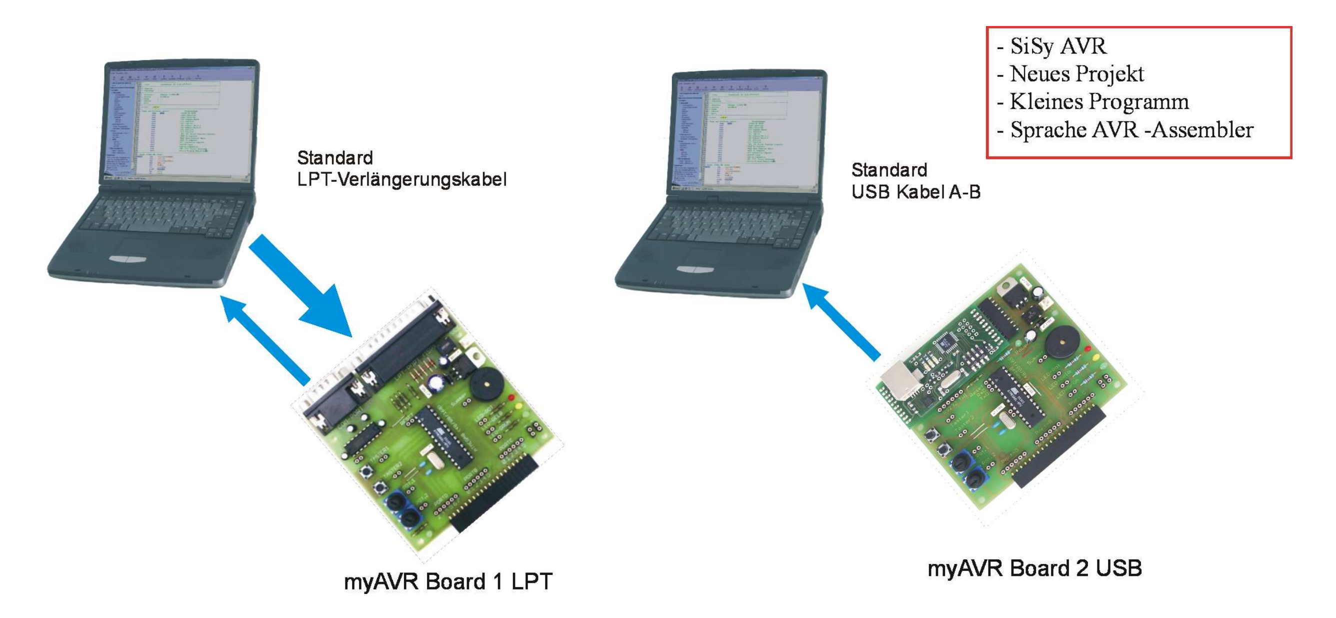

Compilieren und Linken

Der eingegebene Quellcode muss nun in Maschinencode für den AVR-Prozessor übersetzt werden. Wählen Sie dazu die Schaltflächen „Kompilieren“ und „Linken“. Bei fehlerfreier Übersetzung liegt das Programm als „Lauflicht.hex“ vor und kann auf den FLASH-Programmspeicher des Prozessors gebrannt werden.



Hardware anschließen und brennen

Das myAVR Board verfügt über eine ISP (In System Programming) Schnittstelle. Der Prozessor muss also nicht für die Programmierung aus dem System entfernt werden, um ihn in einem gesonderten Programmiergerät zu brennen, sondern kann im myAVR Board direkt programmiert werden. Dazu schließen Sie entsprechend Ihrer Boardvariante das Programmierkabel an den LPT- oder USB-Port Ihres Rechners an.



Zum Brennen wählen Sie die Schaltfläche „Brennen“. Bei erfolgreichem Brennvorgang erhalten Sie im Ausgabefenster folgende Meldung:

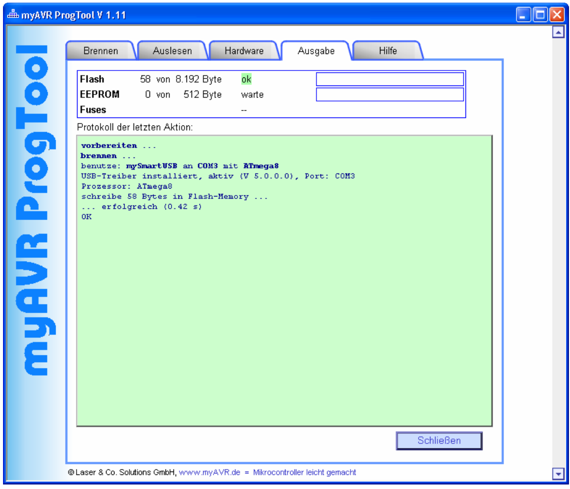


Abbildung 7: Ausgabefenster

Mikrocontrollerlösung testen

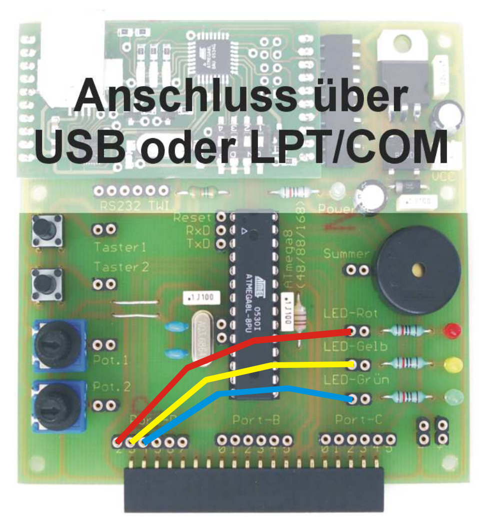


Abbildung 8: Verkablung

Um das Programm zu testen ist es nötig, den Port D mit den Ausgabegeräten LED zu verbinden.

- Wenn vorhanden, ziehen Sie die Batterie bzw. das Netzteil und das Programmierkabel ab.
- Verbinden Sie die LEDs mit dem Prozessorport D entsprechend dem folgenden Schema.
  Nutzen Sie Patchkabel!
- Prüfen Sie die Verbindungen und schließen Sie die Batterie/das Netzteil oder das Programmierkabel wieder an und nehmen Sie die Mikrocontrollerlösung in Betrieb.
- Es ist jetzt an den LED's ein Lauflicht zu sehen.
- Gratulation!
  Das ist Ihre erste Mikrocontrollerlösung mit dem myAVR Board.

Beim Kompilieren, Linken und Brennen des Schnellstart-Beispiels öffnet sich ein Ausgabefenster und zeigt Protokollausgaben der Aktionen an. Beim Brennen öffnet sich zusätzlich das myAVR ProgTool. Wenn die Hardware ordnungsgemäß angeschlossen, von der Software erkannt und das Programm erfolgreich auf den Programmspeicher des Mikrocontrollers übertragen wurde, schließen Sie das myAVR ProgTool. Die letzte Ausschrift hat folgenden Inhalt:

```
brenne Daten neu
Ende.
```

Abbildung 9: Ausgabefenster mit "Brenn"-Protokoll

Inbetriebnahme, Test und Datenkommunikation mit der Mikrocontrollerlösung erfolgen über das myAVR Controlcenter. Dabei wird über die Schaltfläche „Start“ das Board mit der nötigen Betriebsspannung versorgt und der Controller gestartet. Der Datenaustausch mit dem myAVR Board ist möglich, wenn das Null-Modemkabel an Rechner und Board angeschlossen ist, sowie die Mikrocontrollerlösung dafür vorgesehen ist. Es können Texte und Bytes (vorzeichenlose ganzzahlige Werte bis 255) an das Board gesendet und Text empfangen werden. Die empfangenen Daten werden im Protokollfenster angezeigt. Vergleichen Sie dazu den folgenden Abschnitt zum myAVR-Controlcenter.

6 Das myAVR Controlcenter

6.1 Einleitung

Das myAVR Controlcenter ist ein universelles Terminalprogramm zur Kommunikation mit dem myAVR Board und anderen Mikrocontrollerapplikationen, die über eine serielle Schnittstelle (UART) oder USB Anbindung mit virtuellem COM-Port zum PC verfügen. Es kann für Test- und Debug-Meldungen sowie Visualisierung und Protokollierung von Messdaten genutzt werden. Dazu bietet das myAVR Controlcenter umfangreiche Konfigurationsmöglichkeiten.

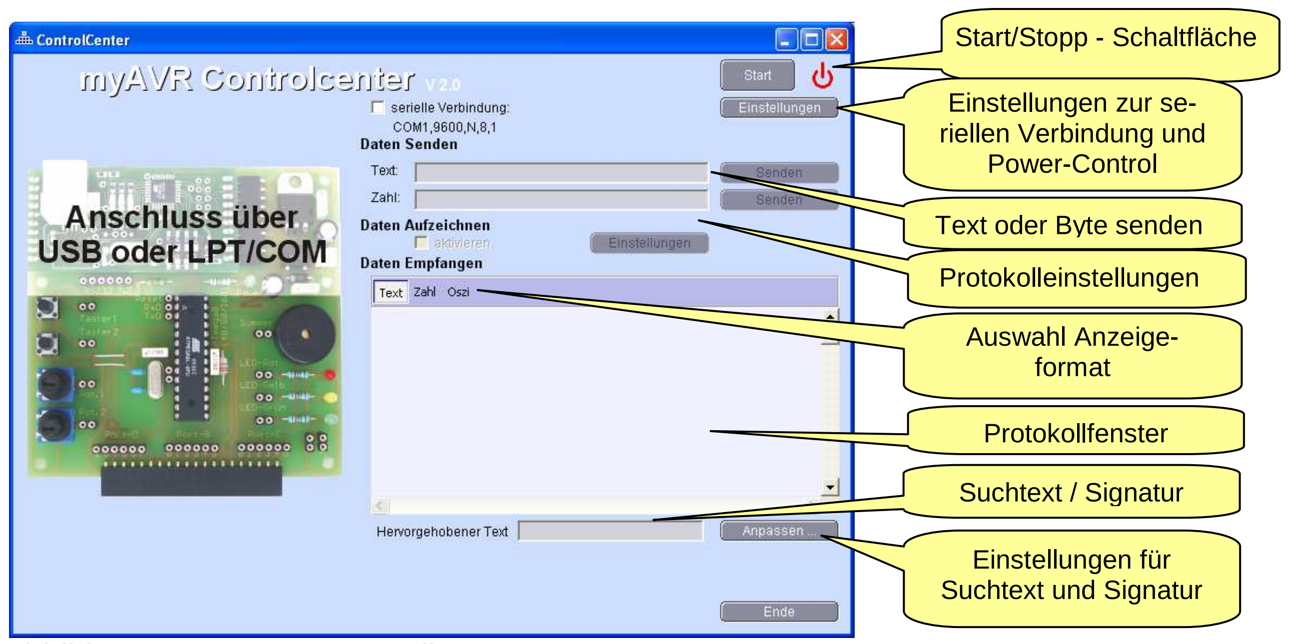
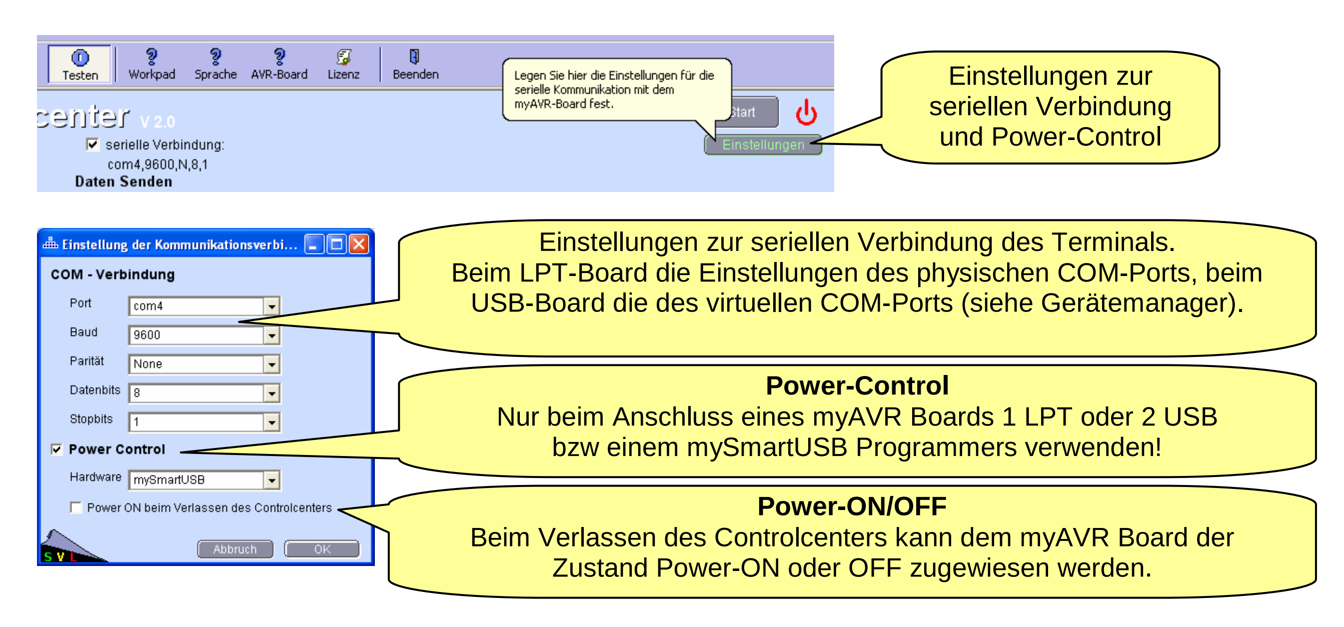


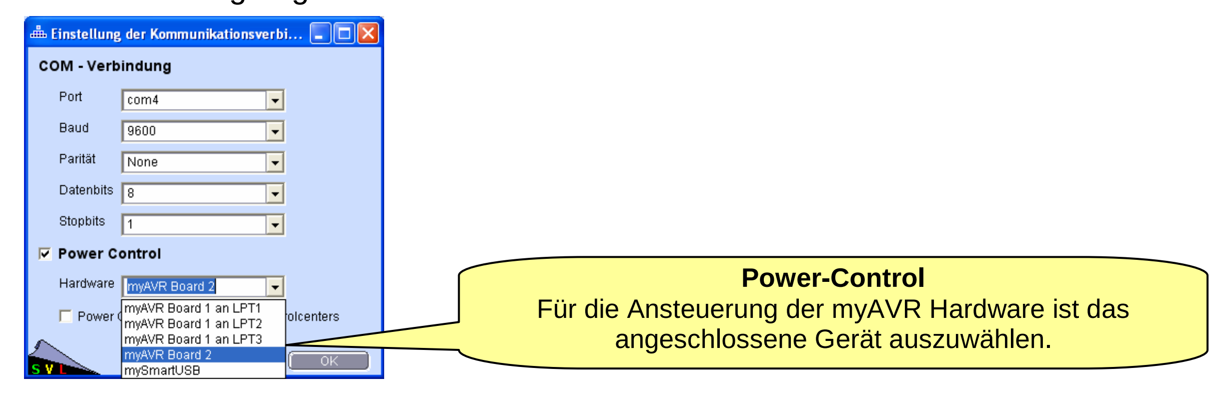
Abbildung 10: myAVR Controllcenter

6.2 Das myAVR Board starten und stoppen (Power Control)

Wenn Sie ein myAVR Board Version 1 LPT oder Version 2 USB besitzen, können Sie die Spannungsversorgung des Boards aus dem Controlcenter heraus steuern. Dazu wählen Sie die Einstellungen für die serielle Verbindung und Power-Control.



Mit dem Aktivieren der Power-Control-Funktion wird beim Betätigen der Schaltfläche START/STOPP die rechnerseitige Spannungsversorgung des myAVR Boards und die RESET-Leitung angesteuert.

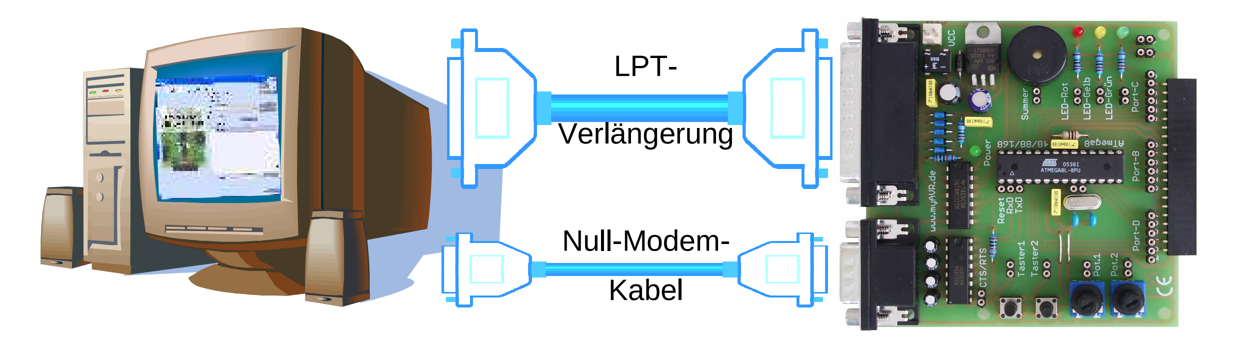


Bei einer externen Spannungsversorgung, zum Beispiel durch einen 9V Block oder das myAVR Netzteil, hat die Power-Control-Funktion keine Wirkung auf die Boardspannung. In diesem Fall bezieht sich die Wirkung ausschließlich auf die RESET-Leitung, um den Controller zu starten bzw. zu stoppen. Beachten Sie, dass beim myAVR Board 1 LPT die Spannungsversorgung über den LPT-Port für eine serielle Kommunikation und den Betrieb des LC-Displays nicht ausreicht, dafür wird die Verwendung eines myAVR Netzteils empfohlen.

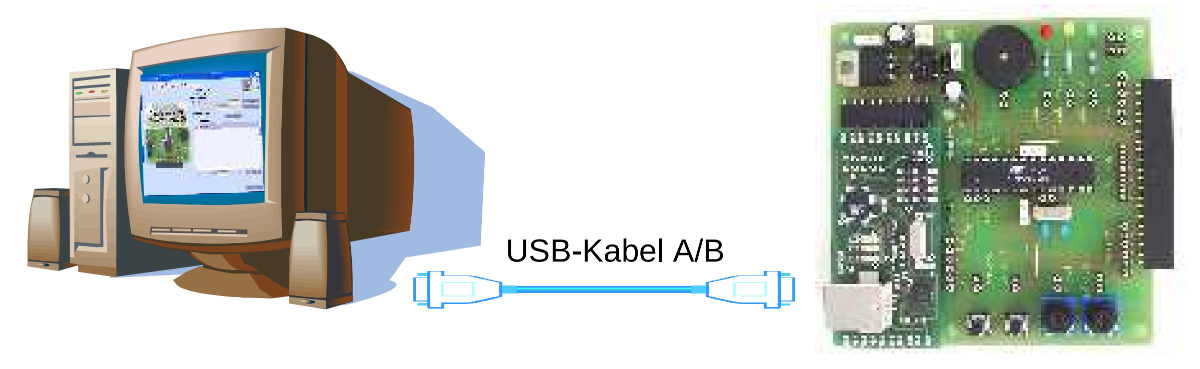
6.3 Kommunikation mit dem myAVR Board

6.3.1 Grundlagen (LPT/USB-Variante)

Die zwei myAVR Boardversionen unterscheiden sich grundsätzlich in der technischen Realisierung der seriellen Kommunikation. Das myAVR Board 1 LPT verfügt über eine gesonderte RS232 Schnittstelle (COM-Port) und wird über ein Nullmodemkabel an einen physisch vorhandenen COM-Anschluss des PC angeschlossen.

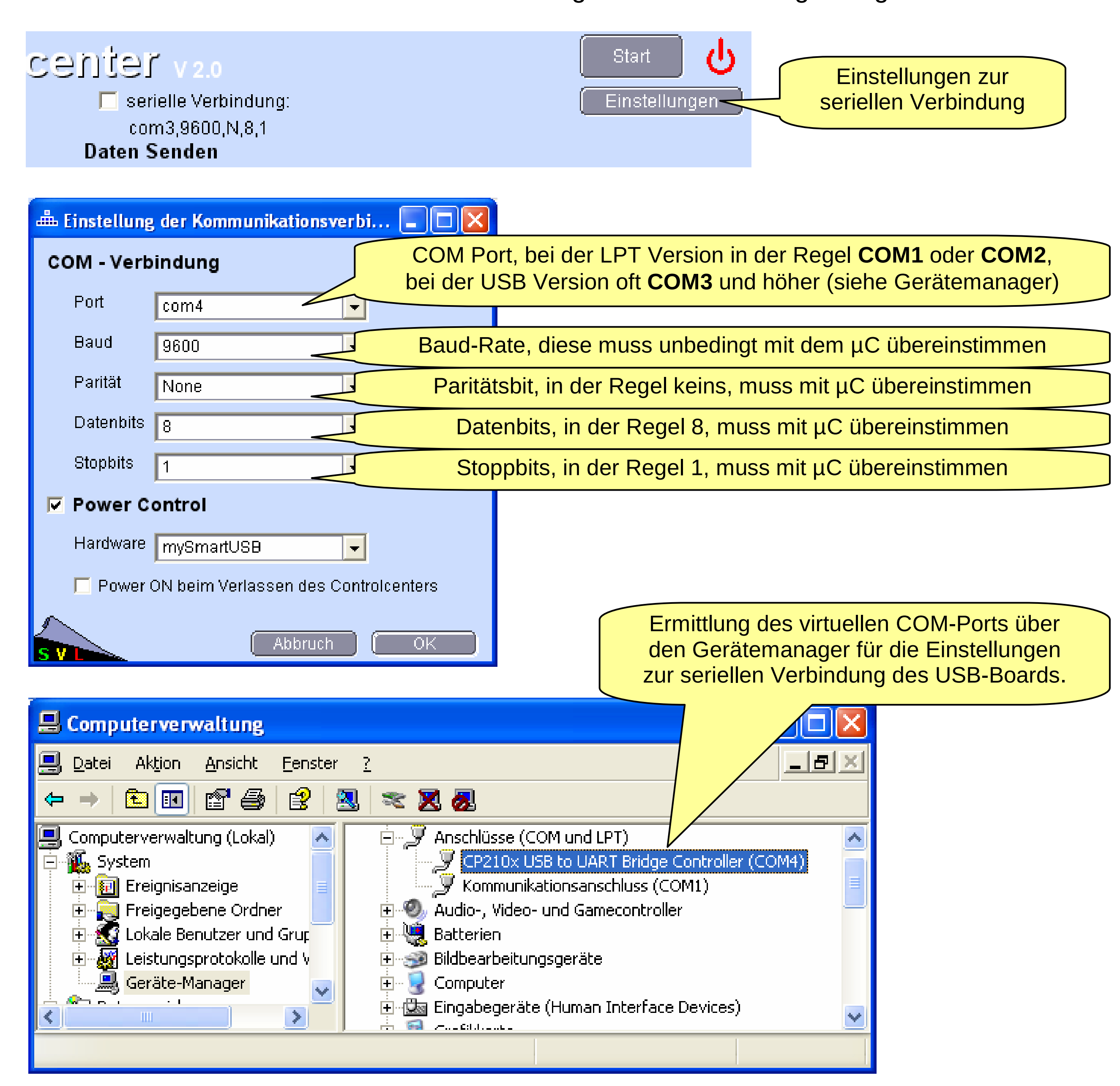


Das myAVR Board 2 USB verfügt über den USB Programmer mySmartUSB. Dieser stellt gleichzeitig einen virtuellen COM-Port für die Kommunikation zur Verfügung.



6.3.2 Einstellungen für die seriellen Verbindung

Für eine erfolgreiche Kommunikation mit dem myAVR Board ist es wichtig, dass Sender und Empfänger von seriellen Daten die gleichen Parameter für die Datenübertragung konfiguriert haben. Auf der PC Seite werden die Kommunikationsparameter im Controlcenter über die Schaltfläche für die Einstellungen der Verbindung konfiguriert.



Im Mikrocontrollerprogramm sind die gleichen Parameter bei der Initialisierung der UART zu wählen. Beachten Sie die eingestellte Taktquelle (hier Quarz mit 3,6 MHz).

```
#define     F_CPU   3686400
#define     BAUD    9600
#include    <avr\io.h>
//----------------------------------------------------------------------------
// UART initialisieren
void initUART()
{
    sbi(UCSRB,3);    // TX aktiv
    sbi(UCSRB,4);    // RX aktivieren
    UBRRL=(uint8_t)(F_CPU/(BAUD*16L))-1;          // Baudrate festlegen
    UBRRH=(uint8_t)((F_CPU/(BAUD*16L))-1)>>8;    // Baudrate festlegen
}
```

Beispiel für die Konfiguration der UART des Mikrocontrollers (µC) mit 9600 Baud.

6.3.3 Daten empfangen vom myAVR Board

Das myAVR Controlcenter empfängt Daten über den gewählten COM-Port und stellt diese im Protokollfenster dar. Damit können Statusmeldungen, Fehlermeldungen oder auch Messwerte erfasst werden. Die Voraussetzung ist, dass die serielle Verbindung hergestellt (Nullmodemkabel oder USB Kabel), korrekt konfiguriert und aktiviert wurde. Die Kommunikation beginnt mit dem Betätigen der Schaltfläche „Start“ und endet mit dem Betätigen der Schaltfläche „Stopp“. Der Zustand (in Betrieb/Halt) wird über das Symbol (rot/grün, EIN/AUS) rechts neben der Schaltfläche angezeigt.



6.3.4 Darstellung der empfangen Daten

Die empfangenen Daten werden fortlaufend im Protokollfenster dargestellt. Der Darstellungsmodus kann während der Kommunikation umgeschaltet werden. Das Controlcenter bietet folgende Darstellungsmodi:

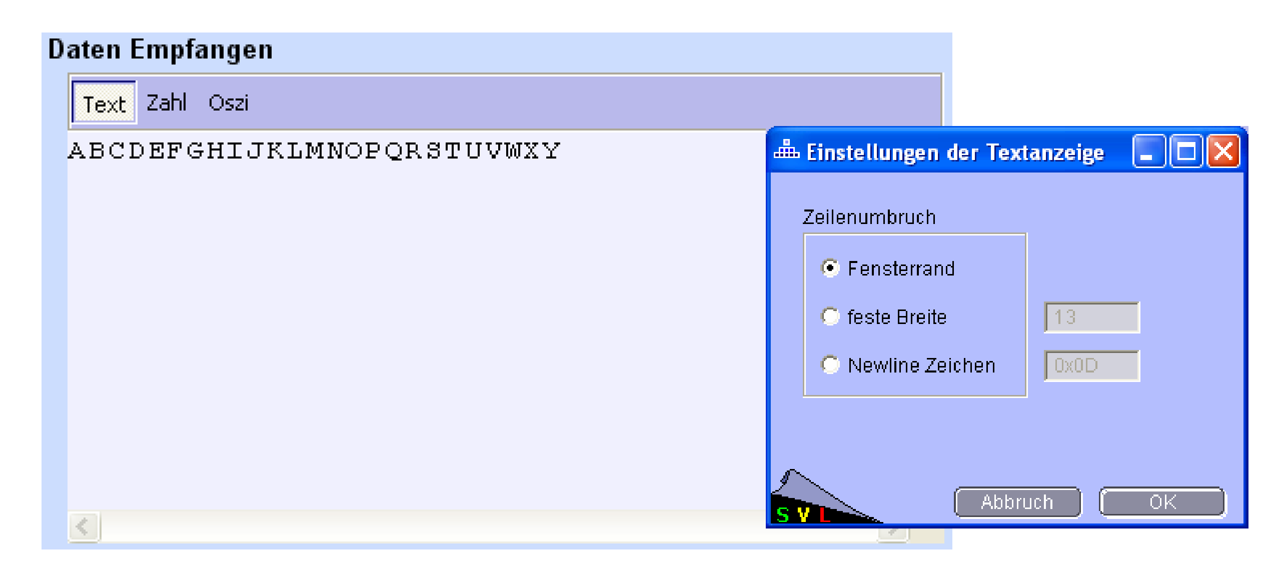
- Text
- Zahl
  - Dezimal
  - Hexadezimal
- Grafik (Oszi)

Die Daten des folgenden Programmbeispiels sollen als Testdaten dienen:

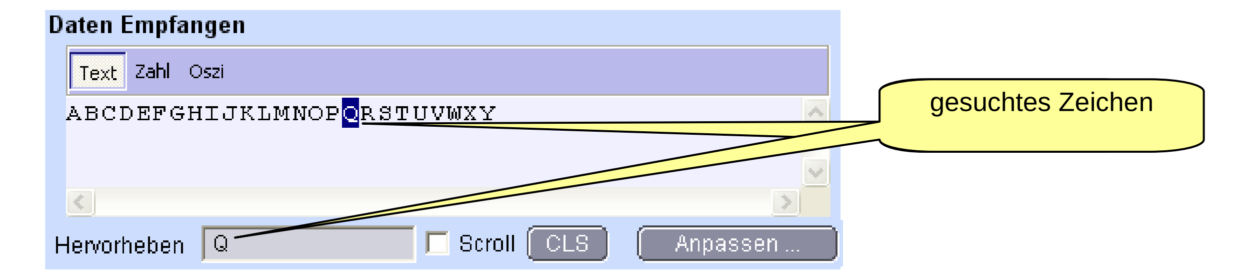
```
//------------------------------------------------------
main()
{
    uint8_t a='A';                  // Zeichen
    initUART();               // Initialisierungen
    while (true)              // Mainloop-Begin
    {
        putChar(a);                 // Zeichen senden
        a++;                        // nächstes Zeichen
        myWait_1000ms();            // Pause
    }                         // Mainloop-Ende
}
//------------------------------------------------------
```


Der Textmodus

Der Textmodus dient zur Visualisierung alphanumerischer Werte im ASCII Format (Zeichenketten). Zahlen, die mit Zeichenketten gesendet werden, müssen vom Mikrocontrollerprogramm zuvor ins ASCII-Format gewandelt werden (siehe `itoa` und `sprintf`).

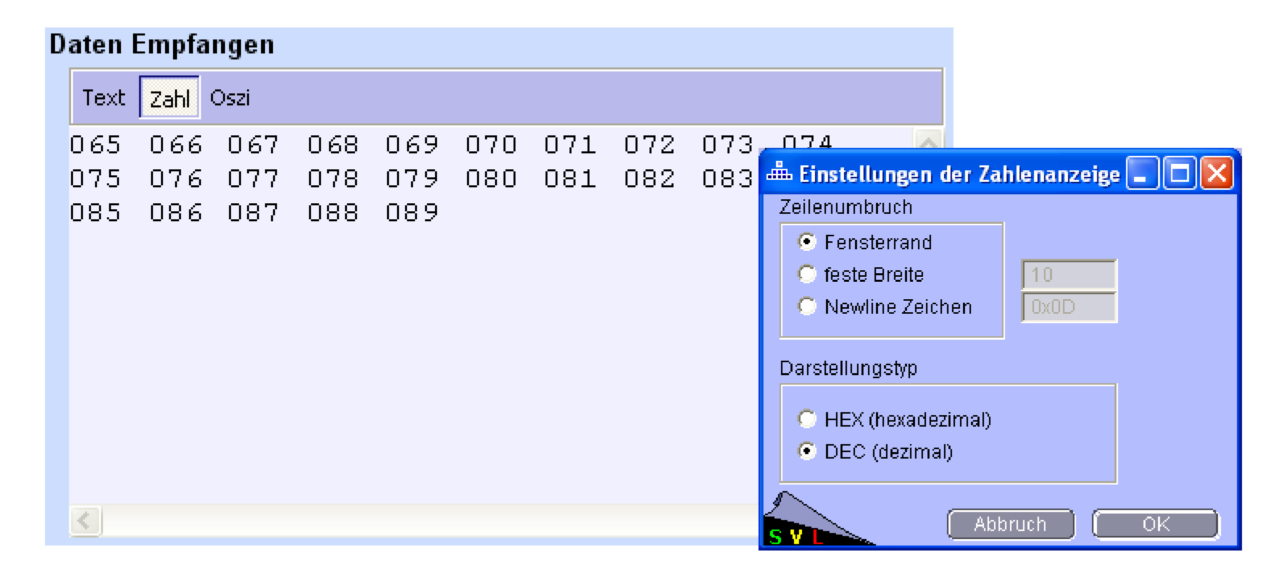


Über die Schaltfläche „Anpassen “ lässt sich das Protokollfenster weiter konfigurieren. Die Zeilenbreite und das Zeichen für einen Zeilenumbruch lassen sich auswählen. Es können wichtige/gesuchte Textstellen im Protokollfenster hervorgehoben werden.

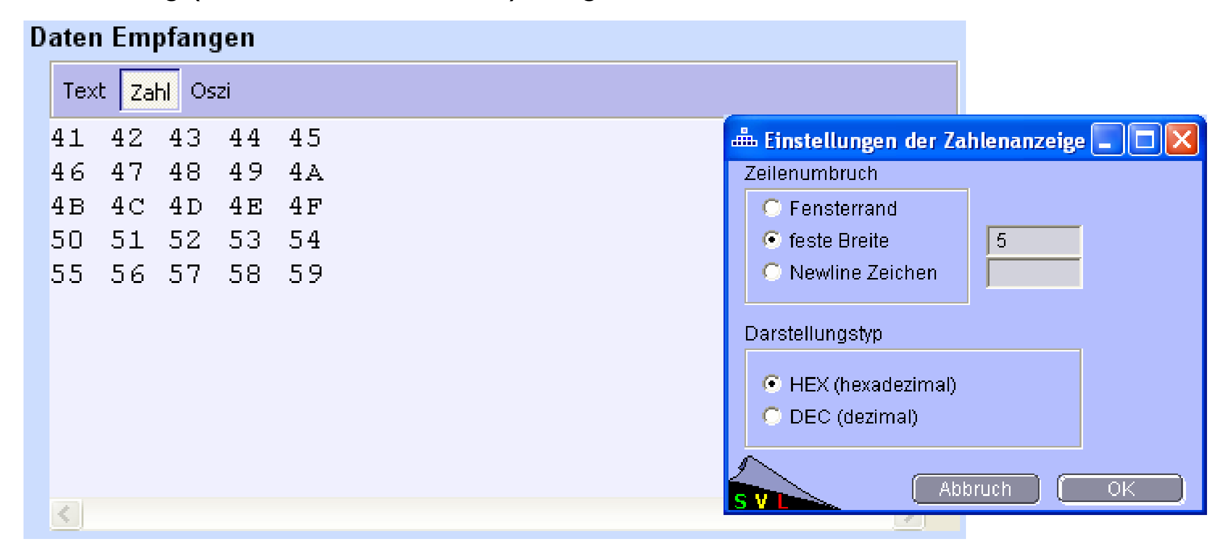


Der Zahlenmodus

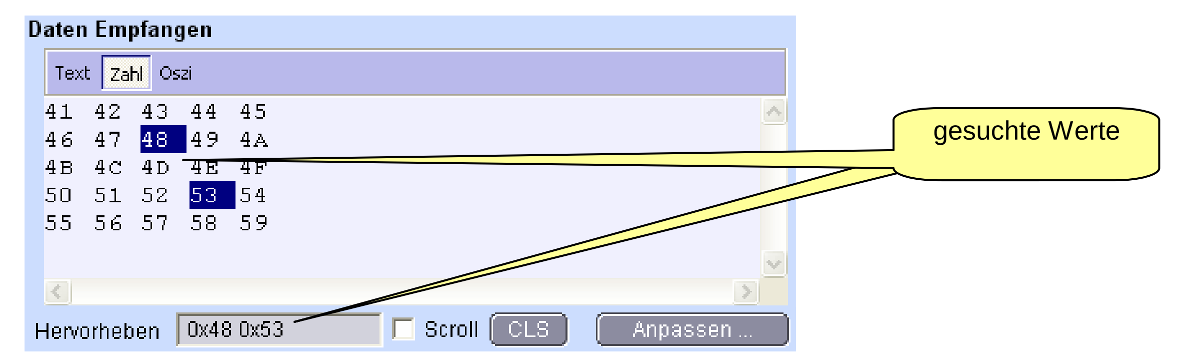
Der Zahlenmodus visualisiert die empfangenen Datenbytes (8 Bit, Werte von 0 bis 255) als Dezimal- oder Hexadezimalzahlen. Dezimalzahlen werden dreistellig mit führender Null dargestellt.



Die Anzeige kann über die Schaltfläche „Anpassen" weiter konfiguriert werden. Dabei werden die Zeilengröße oder das Zeilenumbruchzeichen sowie das Format der Zahlendarstellung (Hexadezimal/Dezimal) ausgewählt.

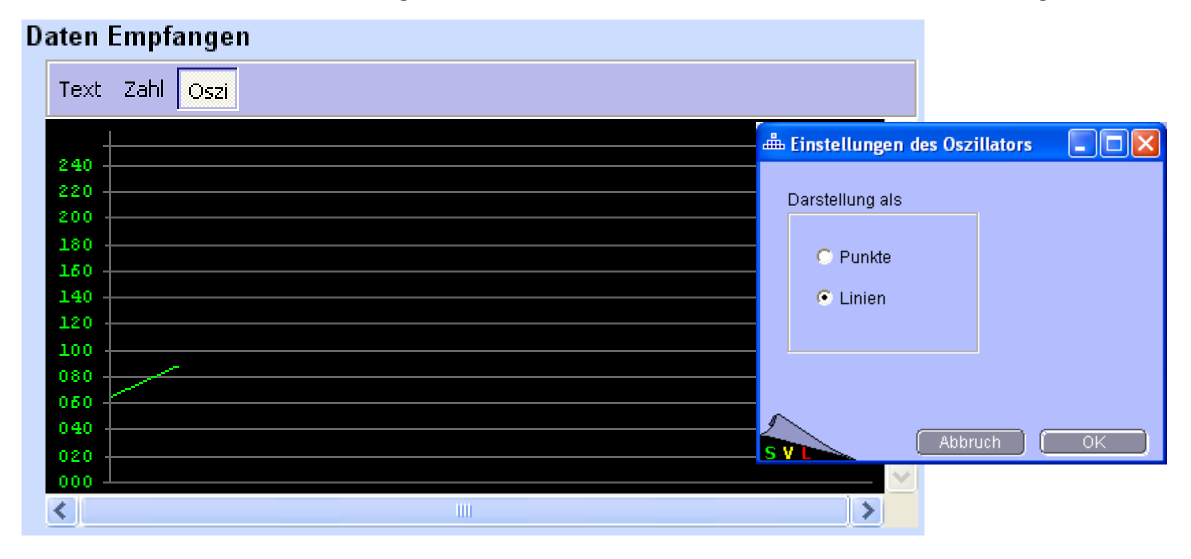


Es können wichtige/gesuchte Zahlenwerte im Protokollfenster hervorgehoben werden. Hexadezimale Zahlen sind durch den Präfix 0x zu kennzeichnen. Mehrere Werte können mit Leerzeichen getrennt angegeben werden.



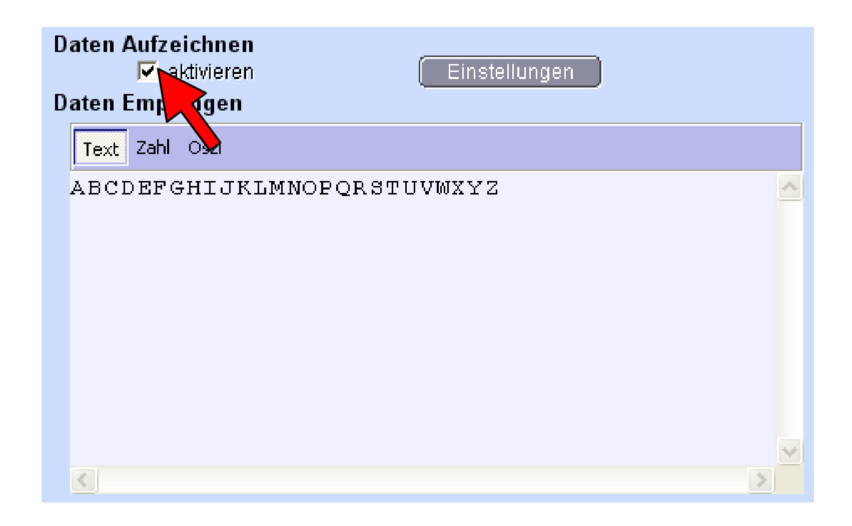
Der Grafikmodus

Messwerte können auch grafisch visualisiert werden. Dabei werden die Werte fortlaufend und byteweise (Wertebereich 0 bis 255) als Punkte in einem Koordinatensystem visualisiert. Die X-Achse repräsentiert den zeitlichen Verlauf, die Y-Achse den Wertebereich der Daten. Die Anzeige kann als einzelne Punkte oder Linien erfolgen.



6.3.5 Empfangene Daten speichern

Bei der Erfassung von Messdaten die vom Mikrocontroller an den PC gesendet werden ist es oft wichtig, diese in eine Datei zu speichern. Damit wird eine Weiterverarbeitung der Daten in entsprechenden Programmen (z.B.: EXCEL) möglich. Das myAVR Controlcenter ermöglicht es, Protokolldaten aufzuzeichnen (Log-Datei, Rekorder-funktion). Um diese Funktion zu aktivieren, wählen Sie die Option „Daten aufzeichnen“



Über die Schaltfläche „Einstellungen“ muss der Dateiname und Pfad der Log-Datei angegeben werden. Zusätzlich ist es möglich, das Format und die Blockgröße (Zeilenumbruch) festzulegen. Die Weiterverarbeitung der Daten erfolgt entsprechend der Möglichkeiten der Zielanwendung (z.B.: Öffnen, Importieren oder Zwischenablage).

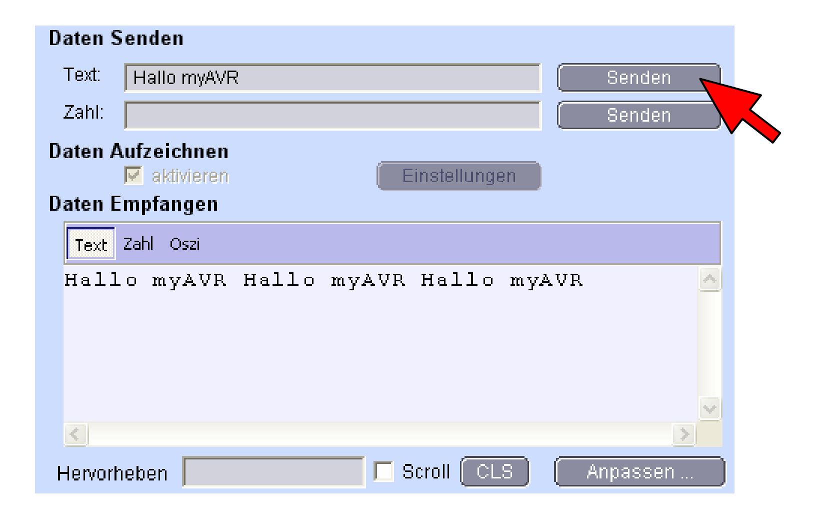
6.3.6 Daten an das myAVR Board senden

Über das Controlcenter können Daten im Text- oder Zahlenformat an den Mikrocontroller gesendet werden. Das folgende Programm dient als Veranschaulichung dieser Funktion.

```
//-------------------------------------------------------
main()
{
    uint8_t zeichen;
    initUART();                 // Initialisierungen
    while (true)                // Mainloop-Begin
    {
        zeichen=getChar();      // Zeichen abholen
        putChar(zeichen);       // Zeichen zurücksenden
    }                           // Mainloop-Ende
}
//-------------------------------------------------------
```

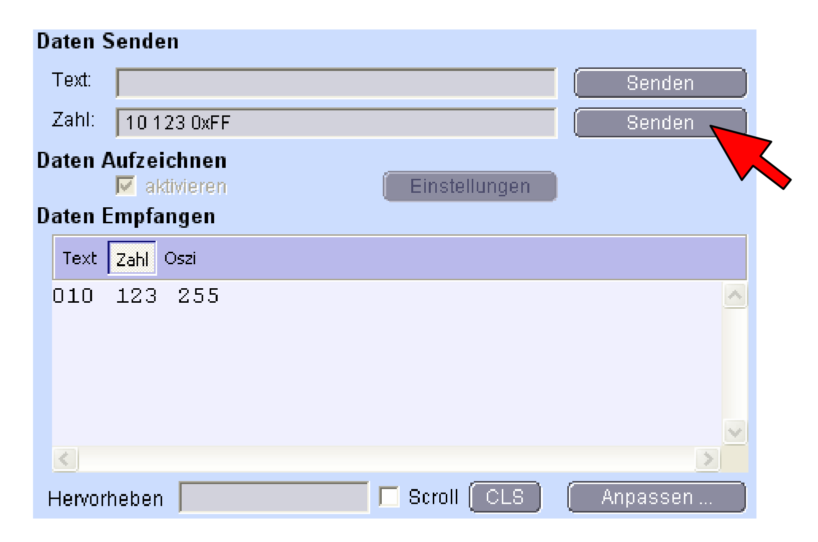
Text senden

Es können einzelne Zeichen, aber auch Zeichenketten gesendet werden. Mit der zugehörigen Schaltfläche „Senden“ wird immer der gesamte Inhalt der Eingabezeile „Text“ gesendet.



Zahlen senden

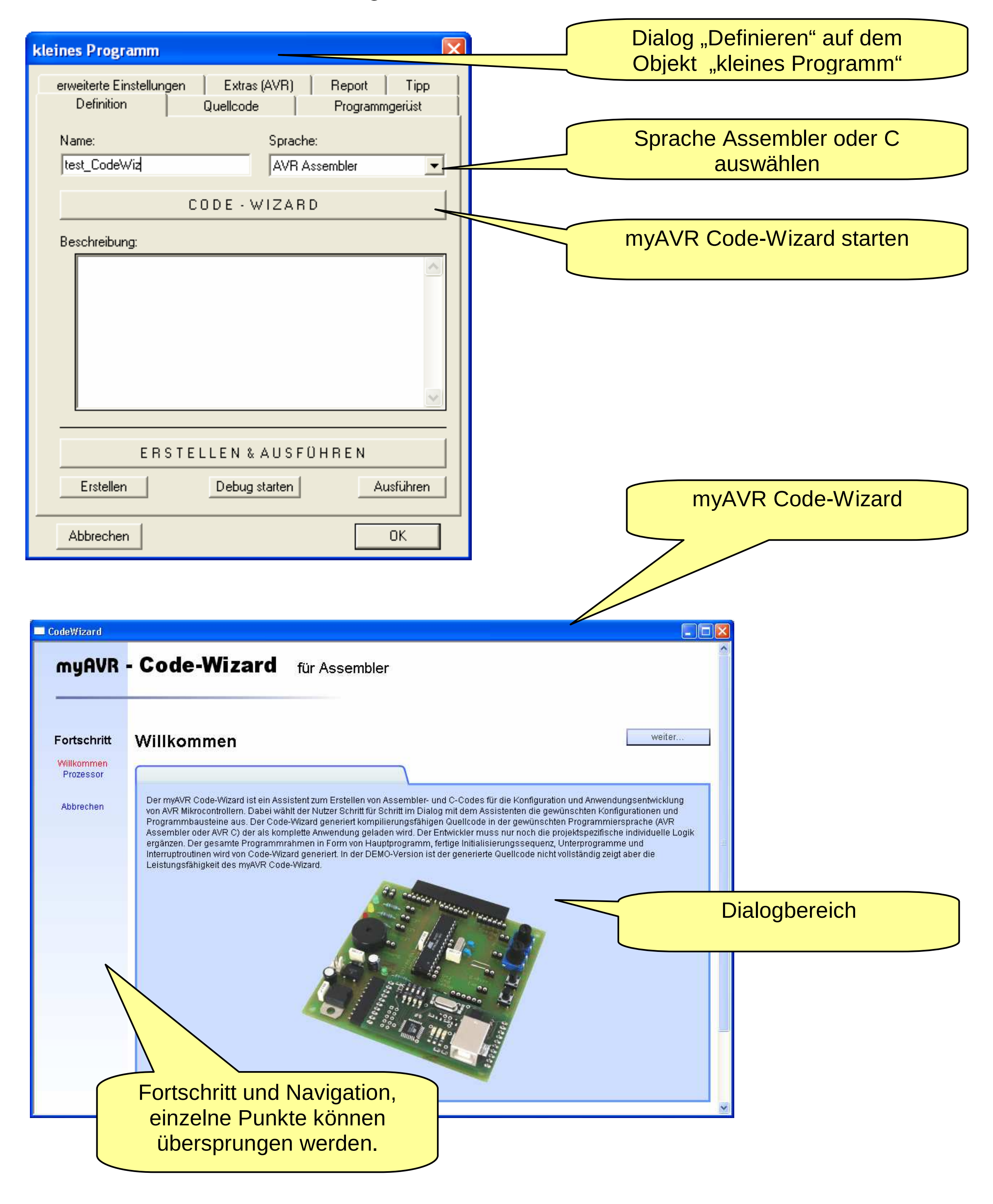
Zahlenwerte von 0 bis 255 (1 Byte) können einzeln oder als Zahlenfolge gesendet werden. Zwischen den Werten ist ein Leerzeichen als Trennzeichen einzufügen. Zahlen, die größer sind als 255, werden auf den niederwertigen Teil gekürzt. Es ist möglich, Zahlen im Hexadezimalformat zu schreiben (Präfix 0x). Mit der zugehörigen Schaltfläche „Senden“ wird immer der gesamte Inhalt der Eingabezeile „Zahl“ gesendet.



7 Der myAVR Code-Wizard

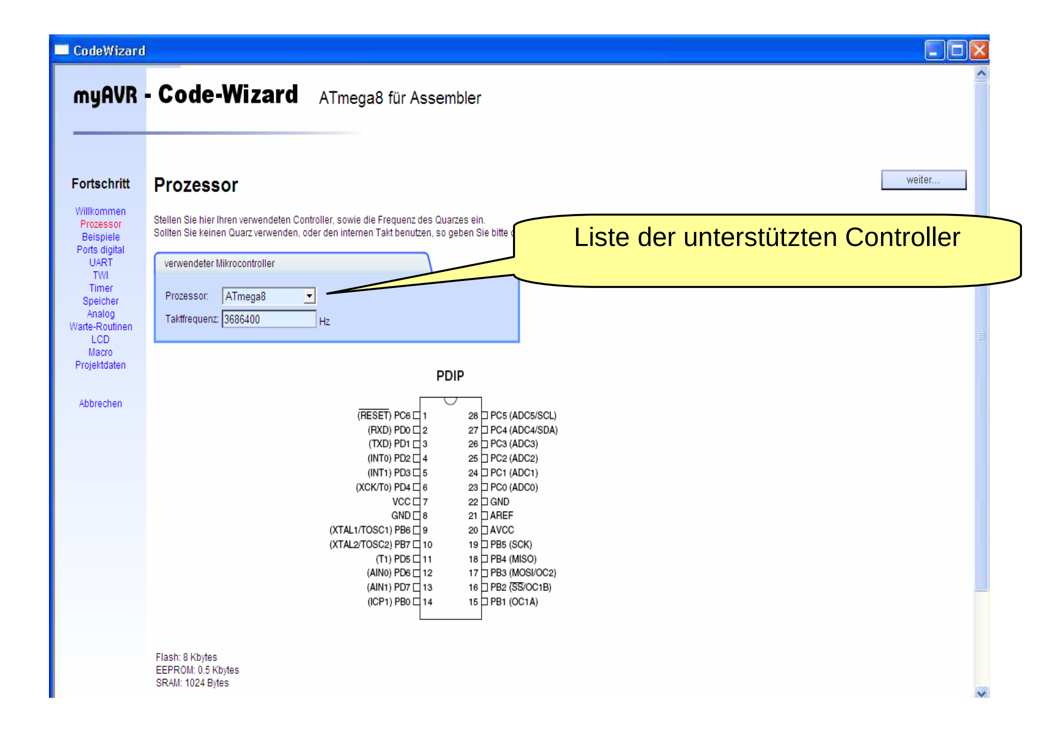
7.1 Einführung

Der myAVR Code-Wizard ist ein Assistent zum Erstellen von Assembler- und C-Codes für die Konfiguration und Anwendungsentwicklung von AVR Mikrocontrollern. Dabei wählt der Nutzer Schritt für Schritt im Dialog mit dem Assistenten die gewünschten Konfigurationen und Programmbausteine aus. Der Code-Wizard generiert kompilierungsfähigen Quellcode in der gewünschten Programmiersprache (AVR C oder AVR Assembler), der als komplette Anwendung geladen wird. Der Entwickler muss nur noch die projektspezifische individuelle Logik ergänzen. Der gesamte Programmrahmen in Form von Hauptprogramm, fertige Initialisierungssequenz, Unterprogramme und Interruptroutinen wird von Code-Wizard generiert.



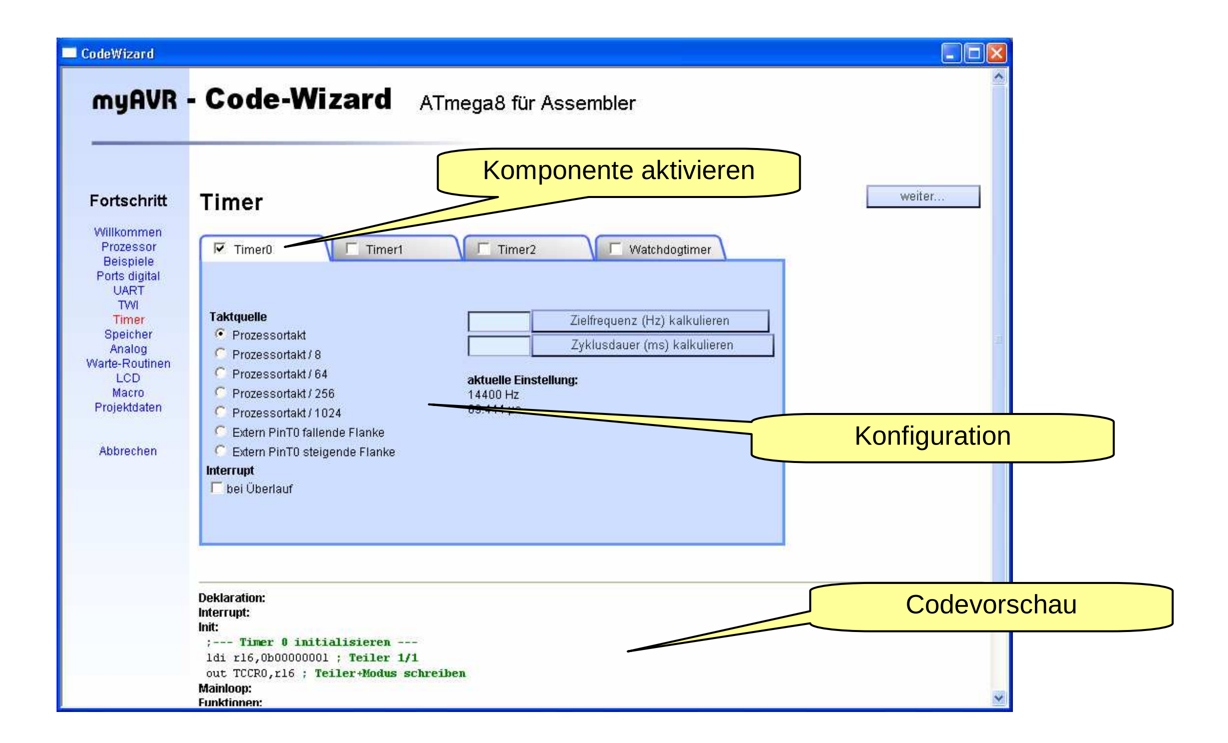
7.2 Grundeinstellungen

Die Zielsprache wurde vor dem Start des Code-Wizards ausgewählt. Für die Generierung von korrektem Quellcode ist es notwendig, den Controllertyp und die Taktgeschwindigkeit festzulegen.



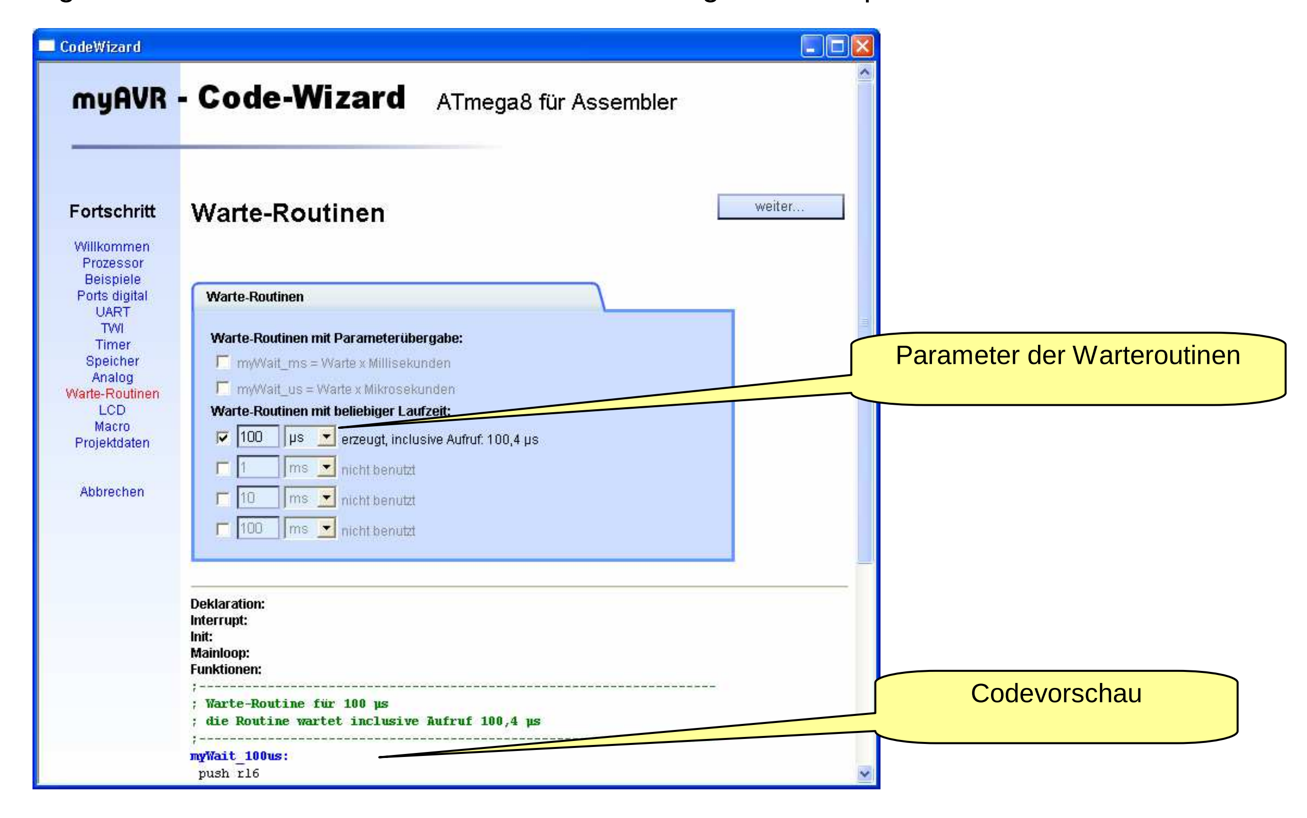
7.3 Geräteeinstellungen

Über den Dialogbereich können Schritt für Schritt die Komponenten des gewählten Controllertyps konfiguriert werden.



7.4 Unterprogramme

Neben der Möglichkeit die Hardwarekomponenten des gewünschten Controllers zu konfigurieren, bietet der Code-Wizard auch eine Reihe typischer Unterprogramme. So verfügt er über einen Warteroutinen-Rechner zum generieren präziser Wartefunktionen.



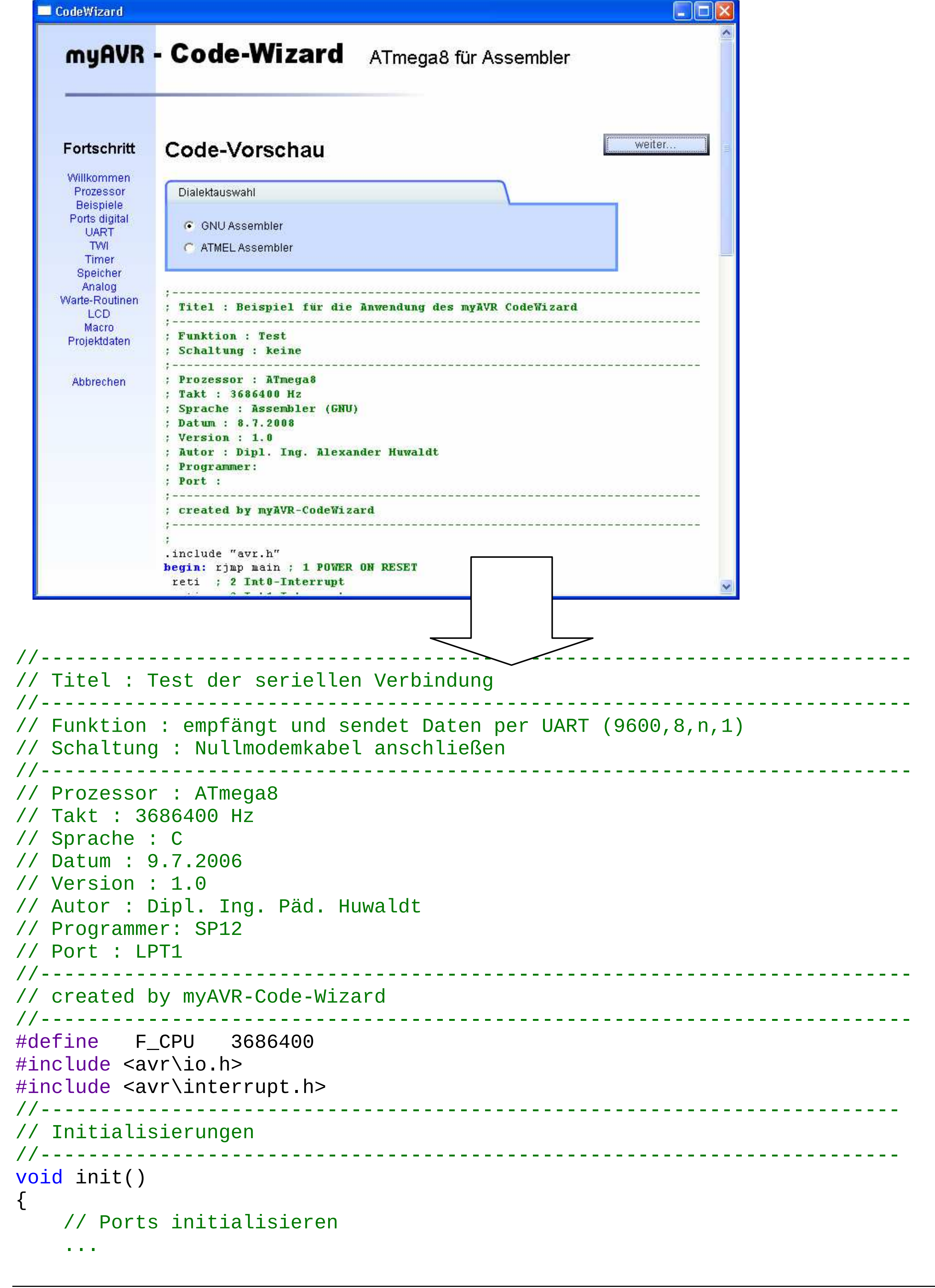
7.5 Projektdaten

Der letzte Punkt im Code-Wizard ist die Eingabe der Projektdaten. Aus diesen wird eine Programmkopfdokumentation generiert und dem kompletten Quellcode vorangestellt.



7.6 Codegenerierung

Der vollständige Quellcode wird zur Kontrolle dem Anwender angezeigt. Es können jetzt noch Änderungen vorgenommen werden, indem man die betreffenden Punkte im Navigationsbereich auswählt und die Parameter ändert. Mit Bestätigung des Quellcodes wird dieser als komplettes Programm eingefügt.




```
//----------------------------------------------------------------------------
// Titel : Test der seriellen Verbindung
//----------------------------------------------------------------------------
// Funktion : empfängt und sendet Daten per UART (9600,8,n,1)
// Schaltung : Nullmodemkabel anschließen
//----------------------------------------------------------------------------
// Prozessor : ATmega8
// Takt : 3686400 Hz
// Sprache : C
// Datum : 9.7.2006
// Version : 1.0
// Autor : Dipl. Ing. Päd. Huwaldt
// Programmer: SP12
// Port : LPT1
//----------------------------------------------------------------------------
// created by myAVR-Code-Wizard
//----------------------------------------------------------------------------
#define   F_CPU   3686400
#include <avr\io.h>
#include <avr\interrupt.h>
//---------------------------------------------------------------------------
// Initialisierungen
//---------------------------------------------------------------------------
void init()
{
    // Ports initialisieren
    ...
```

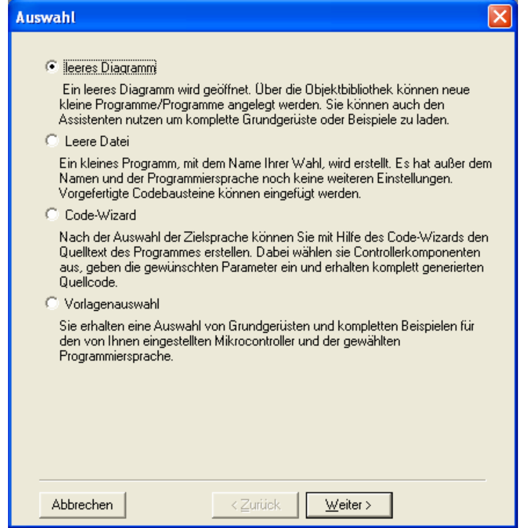

8 Entwicklung eines großen Programms mit SiSy AVR

8.1 Einleitung

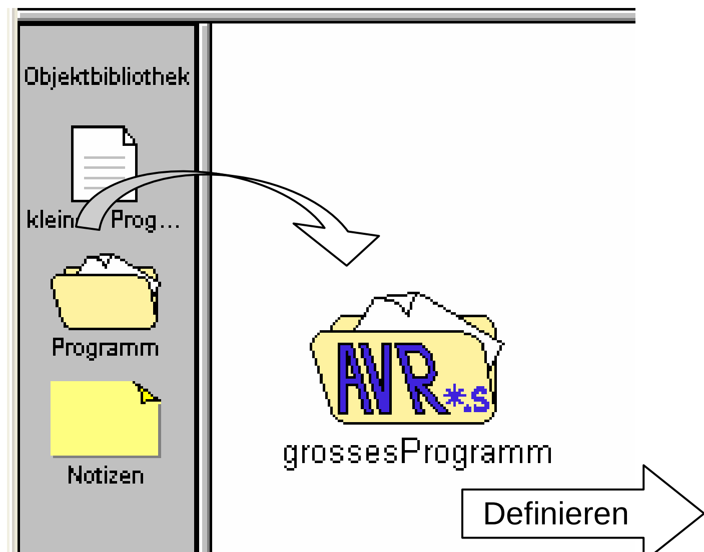
Für die Entwicklung eines größeren Programms ist es unzweckmäßig, alle Befehle in eine Datei (Unit) zu schreiben. Der Grund dafür ist, dass bei mehr als 60 bis 80 Zeilen Quellcode die Übersicht über die Programmstruktur verloren geht. Selbst die Unterteilung in Unterprogramme reicht ab einer bestimmten Größe von Programmen nicht mehr aus. SiSy erlaubt zwar in kleinen Programmen bzw. je Unit 10 Kilobyte Code. Das sind in Assembler zum Beispiel über 1000 Zeilen. Ein kleines Programm bzw. eine einzelne Unit sollte jedoch nie mehr als 80 bis 120 Zeilen haben. Wird diese Grenze erreicht, sollte das Programm in logische Einheiten (Units, Module) gegliedert werden. Dabei fasst man alles zusammen, was zu einer bestimmten Funktionalität oder einer bestimmten Baugruppe wie zum Beispiel dem AD-Wandler gehört. Physisch entstehen dabei mehrere Dateien, die für sich genommen wieder übersichtlich sind da diese dann nur 80 bis 120 Zeilen Code enthalten. Das Übersetzungsprogramm (Assembler, Compiler, Linker) sorgt dann dafür, dass alle einzelnen Units zu einem vollständigen Programm zusammengefügt werden.
Das folgende Kapitel erläutert an einem sehr kleinen Beispiel die Handhabung der Komponenten eines großen Programms, welches in mehrere Units zerlegt wird.

8.2 Vorbereitung

Starten Sie SiSy und legen Sie ein neues Projekt an. Wählen Sie das Vorgehensmodell „Programmierung“. Nehmen Sie die Grundeinstellungen für die verwendete AVR Hardware vor oder lassen Sie die myAVR-Hardware automatisch suchen. Erstellen Sie ein leeres Diagramm!



Ziehen Sie aus der Objektbibliothek ein Objekt vom Typ „Programm“.



Legen Sie Name und Sprache für das Programm fest. Überprüfen Sie gegebenenfalls die Einstellungen unter Extras AVR.



8.3 Aufgabenstellung

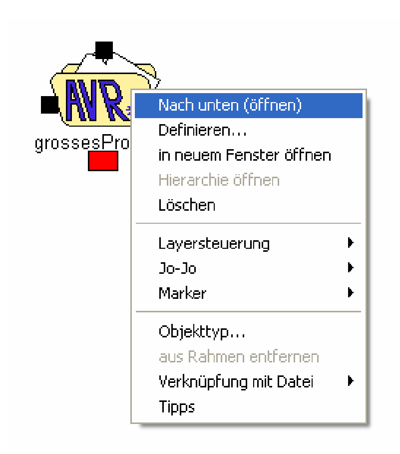
Es ist eine Mikrocontroller-Anwendung mit der Technik des großen Programms (Zerlegen in mehrere Units) zu entwerfen und in der Sprache Assembler zu realisieren.

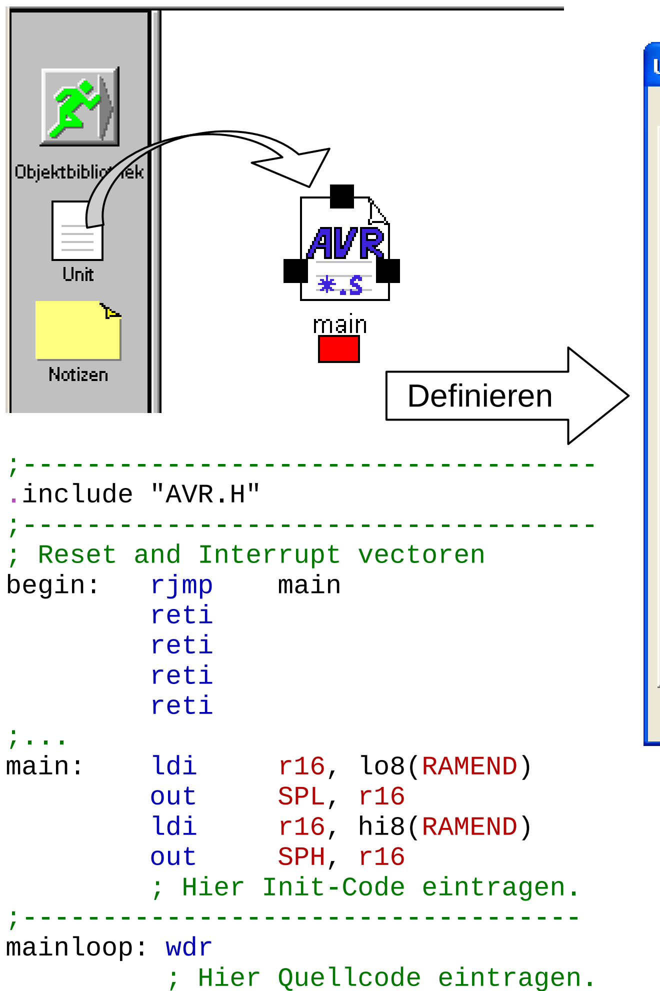
Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Taster eine LED schaltet.

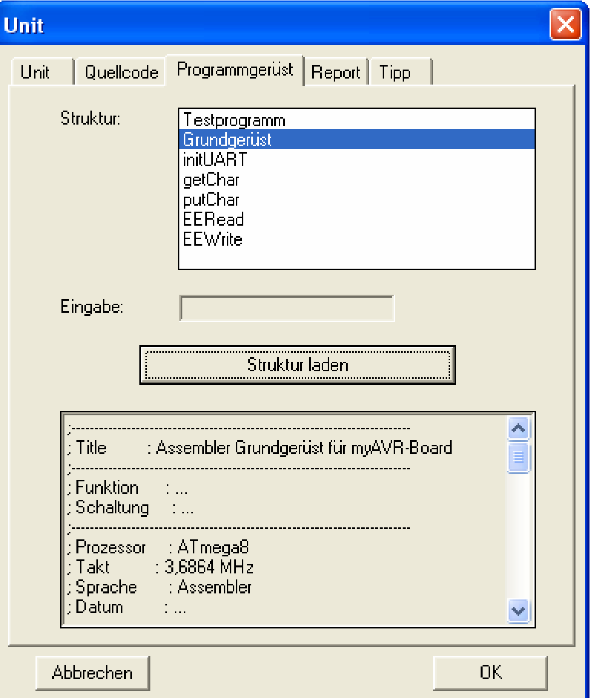
Schaltung:
Port D.2 = Taster 1
Port B.0 = LED

8.4 Hauptprogramm erstellen

Öffnen Sie das Diagrammfenster für ein großes Programm, indem Sie auf dem Symbol rechte Maustaste -> Kontextmenü -> nach unten (öffnen) wählen. Legen Sie eine Unit an. Diese Unit bildet das Hauptprogramm. Nennen Sie die Unit „main“. Damit wird durch die Entwicklungsumgebung erkannt, dass es sich hierbei um das Hauptmodul handelt. Erstellen Sie hier das Hauptprogramm, nutzen Sie die angebotene Vorlage „Grundgerüst“.



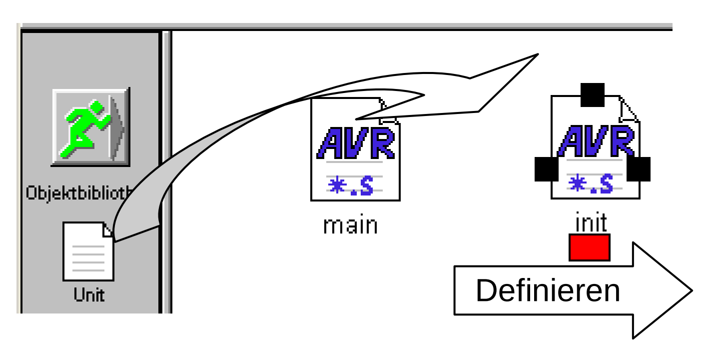


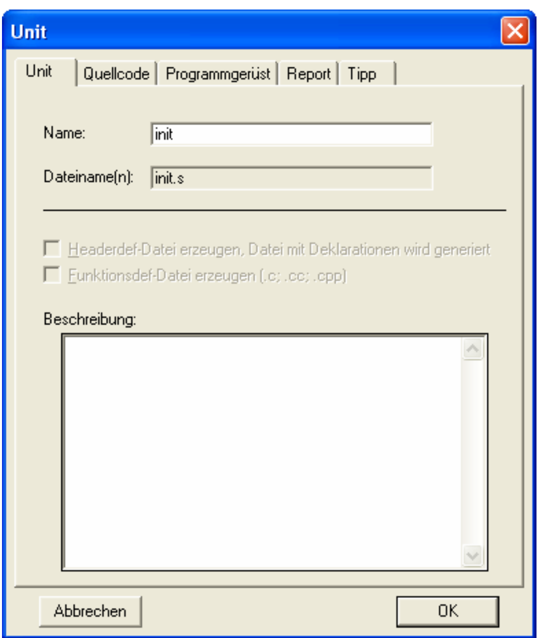



```
;------------------------------------
.include "AVR.H"
;------------------------------------
; Reset and Interrupt vectoren
begin:   rjmp    main
         reti
         reti
         reti
         reti
;...
main:    ldi     r16, lo8(RAMEND)
         out     SPL, r16
         ldi     r16, hi8(RAMEND)
         out     SPH, r16
         ; Hier Init-Code eintragen.
;-----------------------------------
mainloop: wdr
          ; Hier Quellcode eintragen.
          rjmp mainloop
```

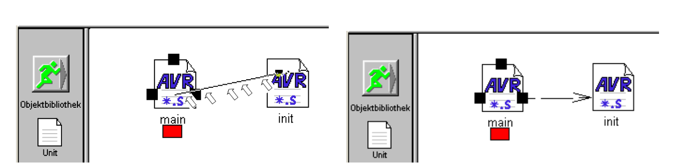

8.5 Units (Unterprogramme) anlegen und verknüpfen

Zur Gliederung des großen Programms wird dieses in mehrere kleine Einheiten (Units/ Module) zerlegt. Diese Einheiten werden nach fachlichen/inhaltlichen Gesichtspunkten gebildet. So kann man alle Initialisierungsaufgaben in der Unit „init" zusammenfassen. Eine Unit kann aus einer Funktion/Unterprogramm oder mehreren Funktionen/Unterprogrammen bestehen. Im einfachsten Fall enthält jede Unit ein Unterprogramm. Legen Sie zusätzlich zur Haupt-Unit „main" die Unit „init" an.



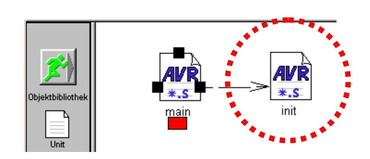


Die Hauptunit benutzt die Unit „init". Daher ist eine Verbindung von der Hauptunit „main" zur Unit „init" zu ziehen. Selektieren Sie die Unit „main" und ziehen vom Verteiler (rot) eine Verbindung auf die Unit „init".



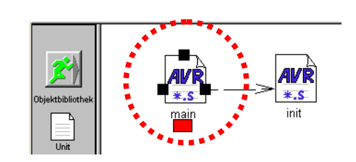
Dabei wird in der Hauptunit „main" ein Include-Eintrag für die Unit „init" erzeugt. Erstellen Sie die Initialisierungsroutine für die benötigten digitalen Ein und Ausgänge.

```
;-----------------------------------------
init:       push    r16
            sbi     DDRB,0    ; B.0 Ausgang
            cbi     PORTB,0   ; B.0 LED aus
            cbi     DDRD,2    ; D.2 Eingang
            sbi     PORTD,2   ; D.2 PullUp
            pop     r16
            ret
;-----------------------------------------
```



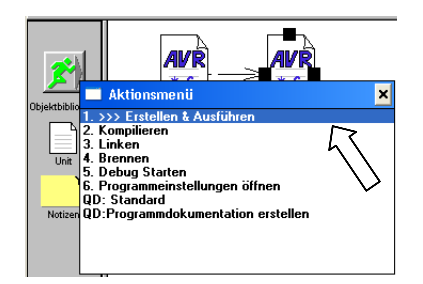
Ergänzen Sie den Code des Hauptprogramms.

```
          ...
          rcall   init
;------------------------------------
mainloop: sbic    PIND,2
          rjmp    led_aus
led_an:   sbi     PORTB,0
          rjmp    mainloop
led_aus:  cbi     PORTB,0
          rjmp    mainloop
;------------------------------------
.include “init.s“
```



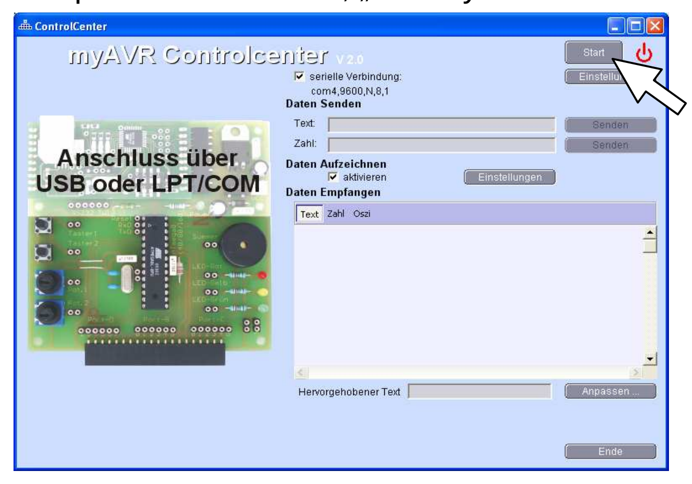
8.6 Übersetzen, Brennen und Test

Zum Übersetzen, Brennen und Testen wählen Sie im Aktionsmenü den entsprechenden Menüpunkt. Im Ausgabefenster erscheint das Protokoll der ausgeführten Aktionen. Des Weiteren öffnet sich für kurze Zeit das myAVR ProgTool.



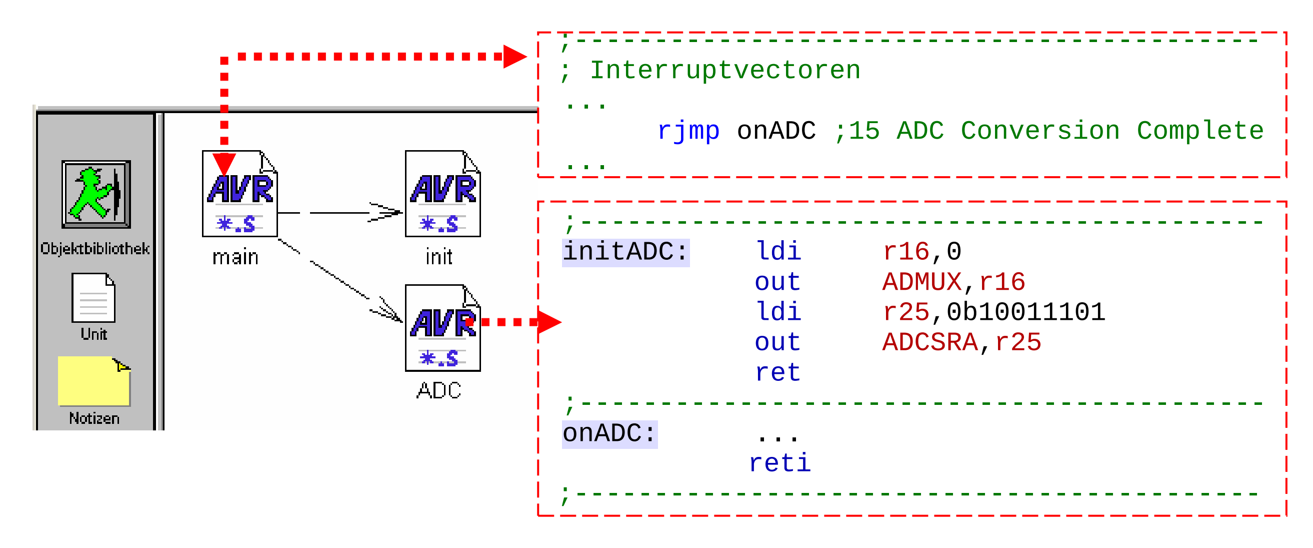
Kompiliere die Datei main.s.
Linke die Datei grossesProgramm.elf.
brenne Daten neu
Öffne myAVR ControlCenter
Ende.

Über das myAVR Controlcenter können Sie das myAVR Board Starten und Stoppen um die Mikrocontrollerlösung zu testen. Überprüfen Sie gegebenenfalls die Einstellungen entsprechend Absatz 6, „Das myAVR Controlcenter“.



8.7 Interrupt-Service-Routine (ISR) im großen Programm

Interrupt-Service-Routinen (im weiteren ISR) sind besondere Formen von Unterprogrammen. Diese werden von einer Interruptquelle des Mikrocontrollers (Timer, ADC, UART, usw.) bei entsprechenden Ereignissen automatisch an beliebiger Stelle im Programmfluss aufgerufen (Unterbrechung, engl. Interrupt). Es ist nötig die Interruptquelle entsprechend zu konfigurieren. Es empfiehlt sich für jedes interruptfähige Gerät eine eigene Unit anzulegen. Diese ist mit der Haupt-Unit zu verbinden.



9 Entwicklung eines Programmablaufplans mit SiSy AVR

9.1 Einleitung

Für die Entwicklung eines Programmablaufplans (PAP) sind konkrete Vorstellungen über die Systemlösung und Kenntnis der Hardware nötig. Ein Programmablaufplan kann aus einer genauen Aufgabenstellung abgeleitet werden.

Beispielaufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Taster eine LED schaltet. Der Controller ist so zu initialisieren, dass an Port B.0 der Taster und an Port B.1 die LED angeschlossen ist. Danach ist fortlaufend der Taster abzufragen. Wenn der Taster gedrückt ist, wird die LED eingeschaltet, sonst bleibt die LED aus.

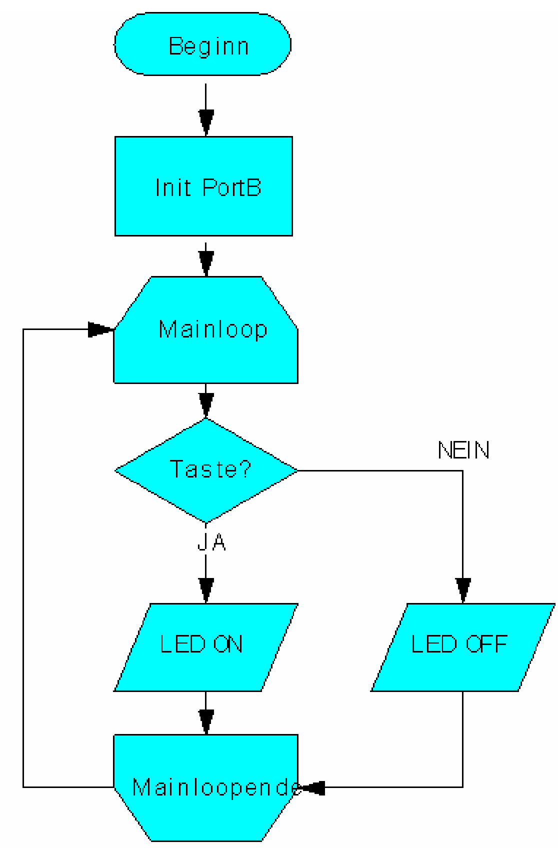


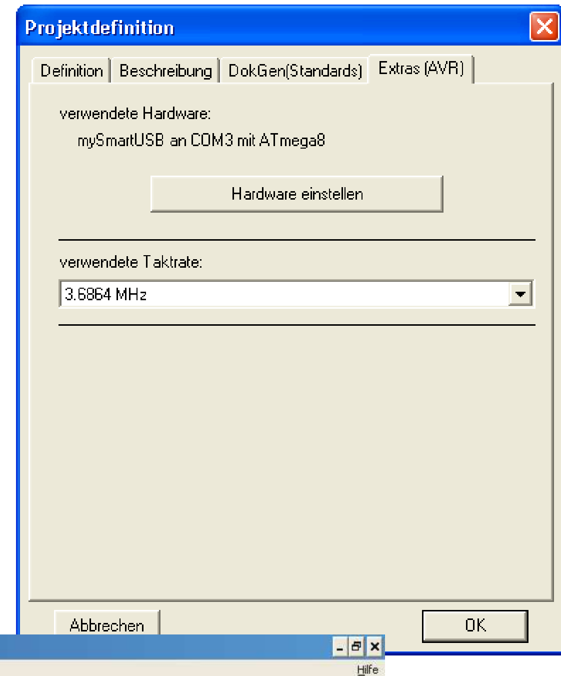
Abbildung 11: PAP zur Beispielaufgabe

9.2 Vorbereitung

Starten Sie SiSy und legen Sie ein neues Projekt an. Wählen Sie das AVR-Vorgehensmodell. Nehmen Sie die Grundeinstellungen für die verwendete AVR Hardware vor oder lassen Sie die myAVR-Hardware automatisch suchen.







Danach öffnet sich die typische Benutzeroberfläche von SiSy mit einem leeren Vorgehensmodell und Sie können mit der Arbeit beginnen. Falls Sie noch die Option „Menü bei Doppelklick“ und „Direkthilfe“ eingeschaltet haben, können Sie diese über den Menüpunkt „Einstellungen“ im Hauptmenü abschalten.

Ziehen Sie als nächstes aus der Objektbibliothek ein Objekt vom Typ „PAP“ in das leere Diagramm. Benennen Sie den PAP mit „Aufgabe1“. Beachten Sie die Einstellungen zum Controllertyp und Programmieradapter unter „Extras (AVR)“; vgl. Abbildung 12.

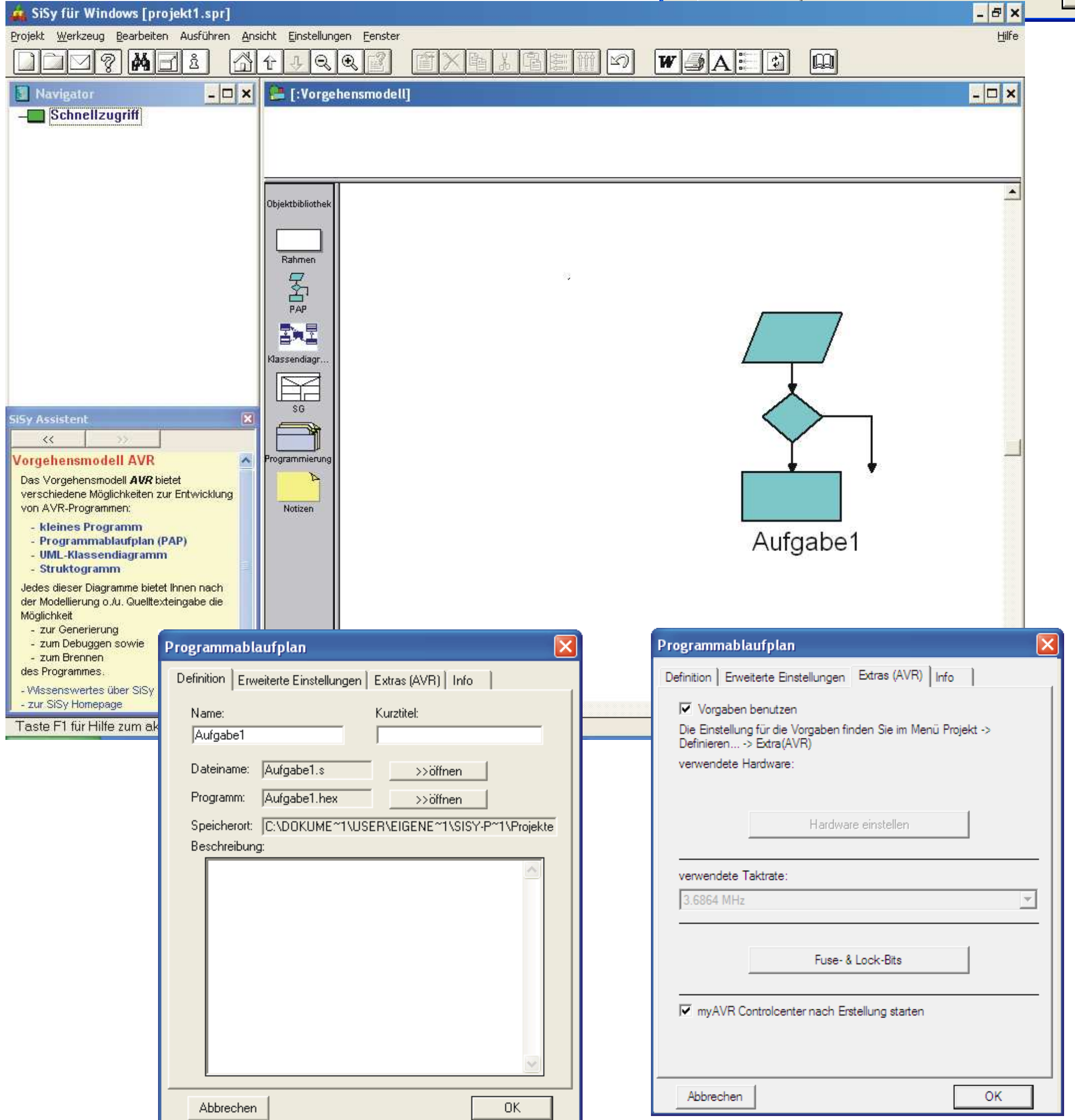


Abbildung 12: Anlegen des Objektes „PAP“ und Einstellungen

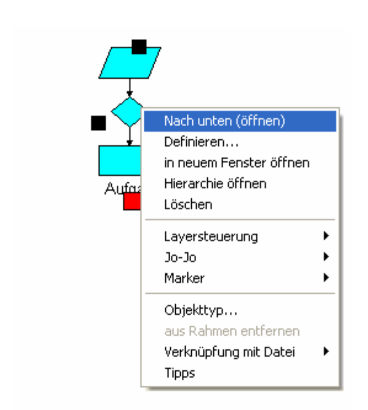


Abbildung 13: PAP öffnen

Der nächste Schritt ist das Aufstellen des Programmablaufplanes. Dazu muss das Diagramm unter dem Symbol geöffnet werden. Wählen Sie rechte Maustaste „nach unten (öffnen)“, um in dieses Diagramm zu gelangen.

9.3 Aufgabenstellung

Es ist im ersten Schritt eine einfache Mikrocontroller-Anwendung mit der Technik des Programmablaufplanes zu entwerfen und in der Sprache Assembler zu realisieren.

Aufgabe:
Entwickeln Sie eine Mikrocontrollerlösung, bei der ein Taster eine LED schaltet.

Schaltung:
Port B.0 = Taster 1
Port B.1 = LED

9.4 Grundstruktur laden

Wenn ein Diagramm leer ist, bietet SiSy typische Vorlagen zum Importieren an, siehe Abbildung 15. Diese können dann weiterbearbeitet werden. Wählen Sie die Diagrammvorlage „Mainprogramm-Grundgerüst“. Abbildung 14 zeigt den PAP zu diesem Grundgerüst.

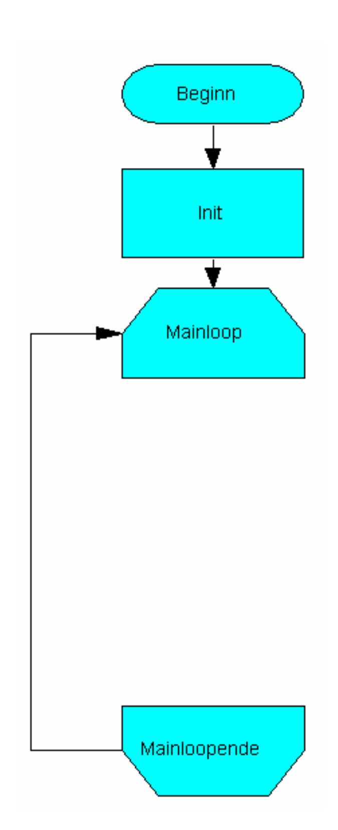


Abbildung 14: Grundgerüst

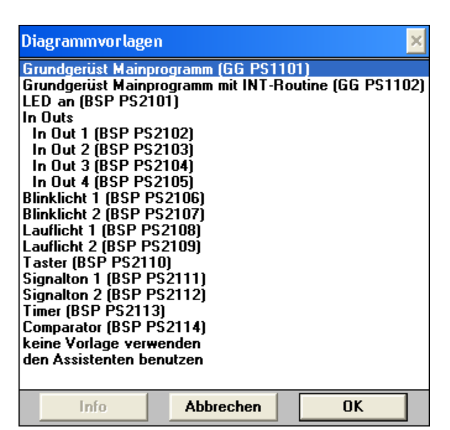


Abbildung 15: Diagrammvorlagen

9.5 Logik entwerfen

Um die Programmlogik im PAP abzubilden, muss die Vorlage um die fehlenden Elemente ergänzt werden. Des Weiteren sind die Elemente durch gerichtete Verbindungen (Kanten) in der Reihenfolge ihrer Abarbeitung zu verbinden.
Um ein Objekt im Diagramm zu ergänzen, wird der entsprechende Objekttyp in der Objektbibliothek mit der Maus ausgewählt und per Drag & Drop an die entsprechende Position im Diagramm gezogen. Objekte können mit dem Mauscursor per Klick selektiert und mittels Drag & Drop auch verschoben werden. Selektierte Objekte lassen sich mit der Taste „Entf" löschen. Verbindungen zwischen den Objekten können über den rot markierten „Verteiler" von selektierten Objekten hergestellt werden. Dazu ist das Ausgangsobjekt zu selektieren mit dem Mauscursor auf den roten Verteiler zu klicken und bei gedrückter linker Maustaste eine Verbindung zum Zielobjekt zu ziehen. Um Objekte zu benennen, können Sie einen Doppelklick auf dem betreffenden Objekt durchführen oder über rechte Maustaste in den Dialog *Definieren* gelangen.

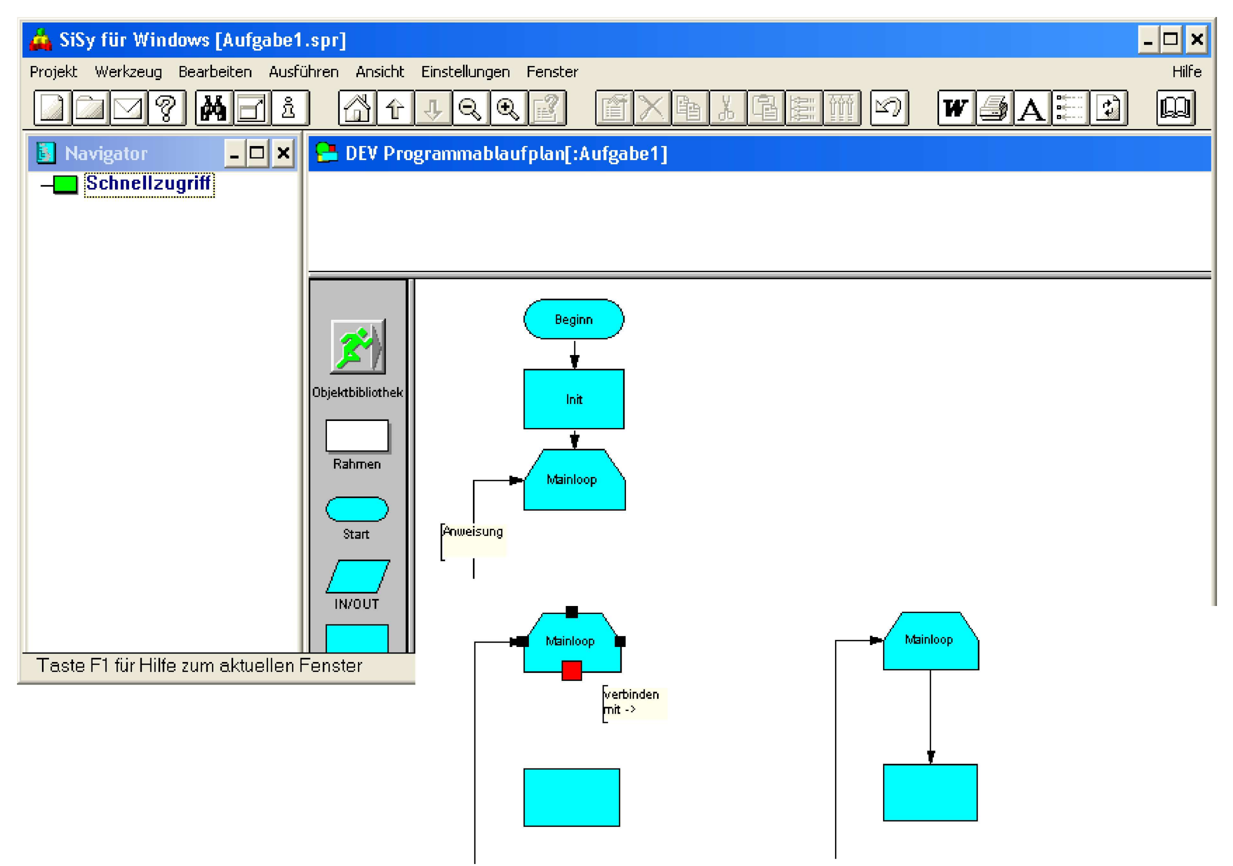


Abbildung 16: Objekte in das Diagramm einfügen und verbinden

Zeichnen Sie den folgenden Programmablaufplan (vgl. Abbildung 17):

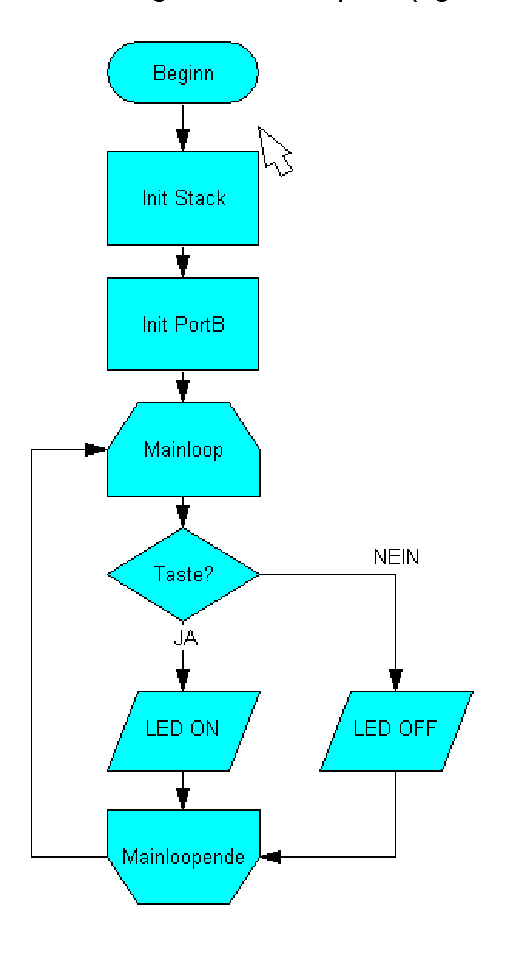


Abbildung 17: Logikentwurf im PAP

9.6 Befehle eingeben

Nachfolgend soll aus dem Programmablaufplan Assemblerquellcode generiert werden. Dazu ist es nötig, die einzelnen Elemente des PAP mit den entsprechenden Assembleranweisungen zu versehen. Dafür gibt es mehrere Möglichkeiten.

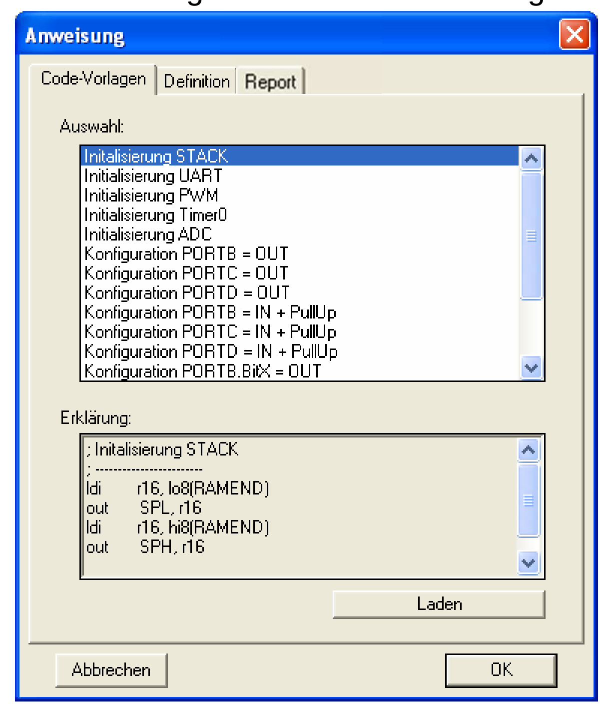


Abbildung 18: Codevorlage PAP

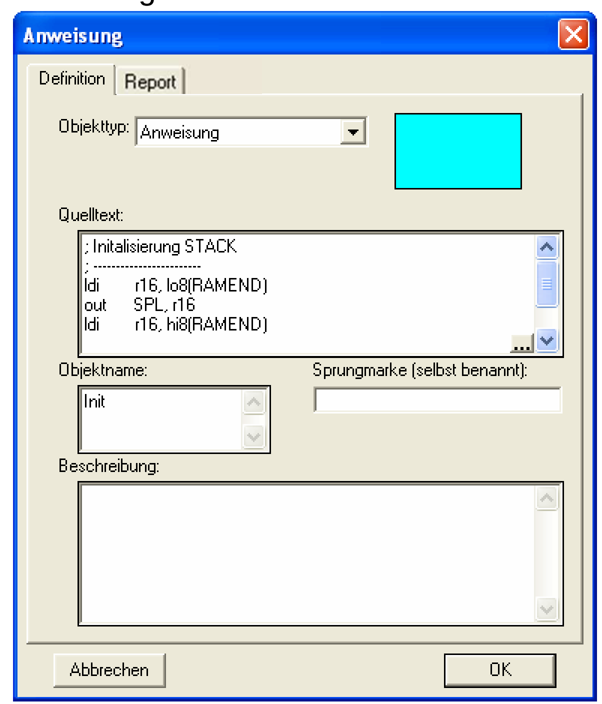


Abbildung 19: Definition PAP-Element

Zum einen bietet SiSy beim ersten Öffnen eines jeden Elementes typische Code-Vorlagen an, die über die Schaltfläche „Laden“ dem Element zugewiesen werden können. Wird der Definieren-Dialog mit der Schaltfläche „OK“ beendet, so wird die Auswahl im Objekt gespeichert und beim nächsten Aufruf des Dialoges „Definieren“ erscheinen die Code-Vorlagen nicht mehr, sondern das Element kann ganz normal bearbeitet werden. In Abbildung 18 und Abbildung 19 sind beide Varianten des Dialoges „Definieren“ zu sehen“.

Die zweite Möglichkeit besteht beim Selektieren von Elementen über den Quellcodeeditor oberhalb des Diagrammfensters, vgl. Abbildung 20 und Abbildung 21.

```
{
0042 ; Initalisierung STACK
0043 ;----------------------------
0044 ldi r16,lo8(RAMEND)
0045 out SPL,r16
0046 ldi r16,hi8(RAMEND)
0047 out SPH,r16
0048
     }
```

Objektbibliothek

Init Stack

Abbildung 20: Quellcodefenster im PAP

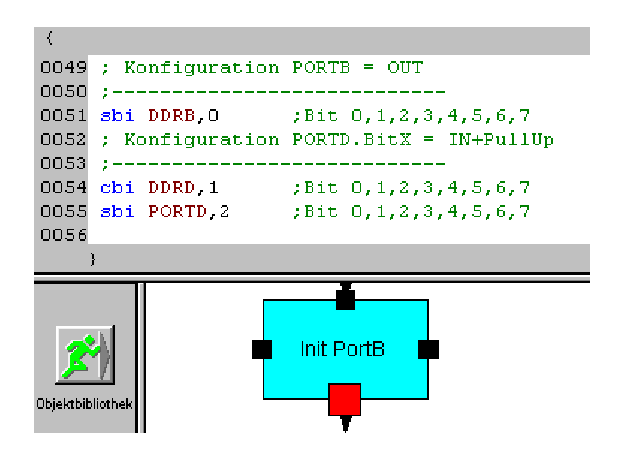


Abbildung 21: Quellcodeeingabe PAP

Geben Sie die gezeigten Quellcodes in die Objekte ein!

Bedingungen haben spezielle Vorlagen, die eine Codegenerierung bei übersichtlichem Programmablaufplan vereinfachen.
Jede Bedingungsvorlage ist so konstruiert, dass eine JA/NEIN Entscheidung erzeugt werden kann. Findet der Codegenerator das Schlüsselwort JA an einer der folgenden Verbindungen, setzt er diese in eine Sprunganweisung „breq“ um. Das Schlüsselwort NEIN wird in „brne“ umgewandelt. Alternativ können statt dieser Schlüsselworte auch der Sprungbefehl selber an eine der Kanten geschrieben werden (breq, brne, brge, brlo, usw.)

Bedingung

Code-Vorlagen | Definition | Report

Auswahl:

Taster an PortB.x gedrückt?
Taster an PortC.x gedrückt?
Taster an PortD.x gedrückt?
Ist Comparator HI/LO?

Erklärung:

```
; Taster an PortB.x gedrückt?
;-------------------------
clz
sbis    PINTB,<x>    ;Bit 0,1,2,3,4,5,6,7
sez                  ;JA = breq
```

Laden

Abbrechen OK

Abbildung 22: Codevorlage Bedingung

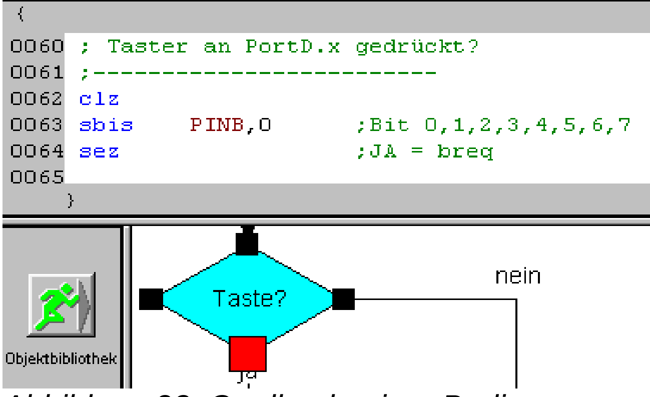


Abbildung 23: Quellcode einer Bedingung

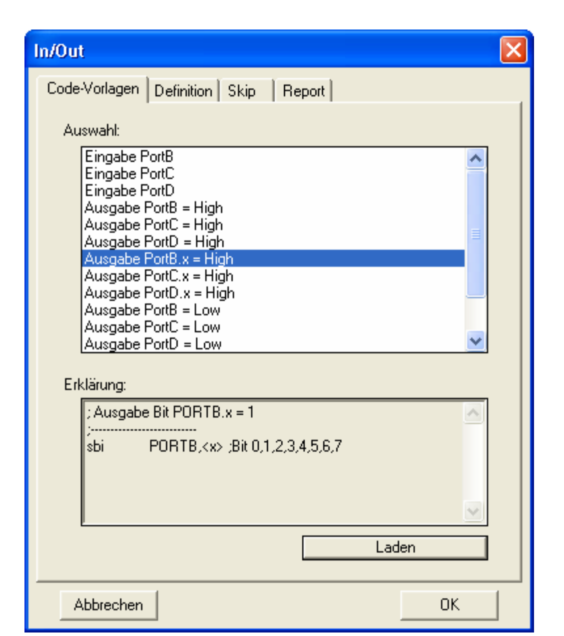
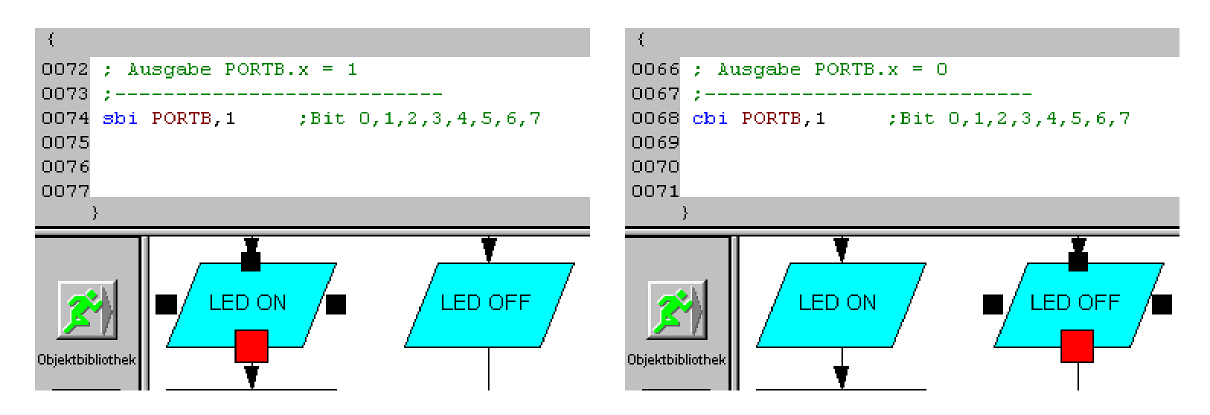


Abbildung 22:Vorlagen für IN/OUT

Auch das Element „IN/OUT“ verfügt über spezifische Vorlagen. Diese sind gegebenenfalls mit zu ergänzen. Dazu sind spitze Klammern als Platzhalter in den Vorlagen eingefügt.

Ergänzen Sie den Quellcode der gezeigten Elemente!



9.7 Übersetzen, Brennen und Test

Sind alle betreffenden Elemente mit Quellcode hinterlegt, kann aus dem Programmablaufplan der komplette Quellcode generiert, kompiliert, gelinkt und auf den Mikrocontroller übertragen werden. Die gewünschte Funktion kann aus dem Aktionsmenü ausgewählt werden (Abbildung 23).

<u>Hinweis:</u>

Beachten Sie, dass für das Brennen des Controllers das Programmierkabel angeschlossen sein muss und bei Bedarf eine geeignete Spannungsquelle anzuschließen ist.

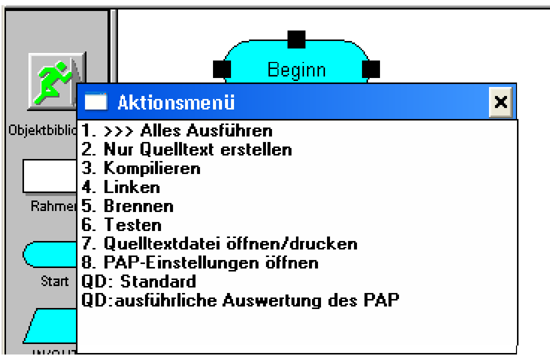


Abbildung 23: Auswahl aus Aktionsmenü

Funktionalitäten des Aktionsmenüs	
Benennung	**Funktion**
Alles Ausführen	Quellcode generieren, kompilieren, linken, brennen
Nur Quelltext erstellen	Quellcode generieren mit allen Marken
Kompilieren	Quellcode zu Objektdatei übersetzen
Linken	Objektdatei zu HEX-Datei binden
Brennen	HEX-Datei an den Controller übertragen
Testen	myAVR Controlcenter öffnen
Quelltextdatei öffnen	Quellcodedatei öffnen
Quelltext bereinigen	Quellcode von überflüssigen Marken bereinigen

Im Ausgabefenster (siehe Abbildung 24) werden die jeweiligen Aktionen angezeigt.

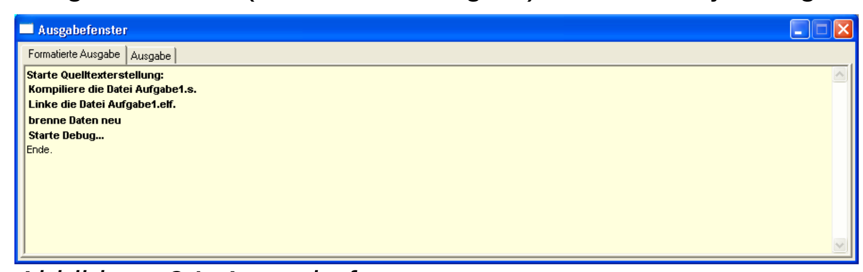


Abbildung 24: Ausgabefenster

Bei Compilerfehlern werden diese ebenfalls im Ausgabefenster mit der entsprechenden Zeilennummer angezeigt. Um zu dem Fehler zu gelangen, genügt meist ein Klick auf die Fehlermeldung. Das betreffende Objekt wird selektiert und die Zeile hervorgehoben.

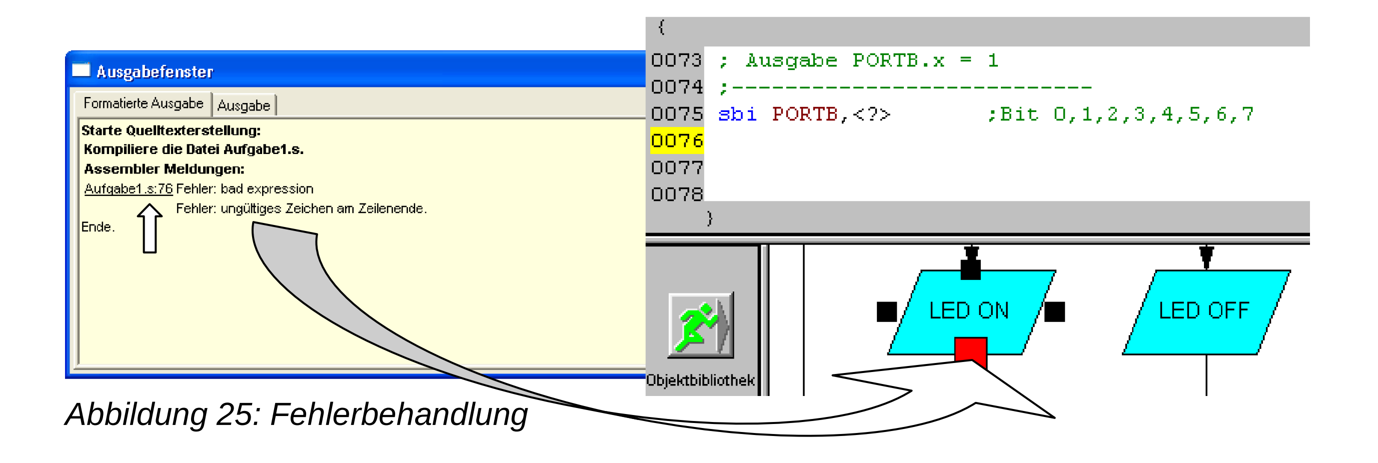


Abbildung 25: Fehlerbehandlung

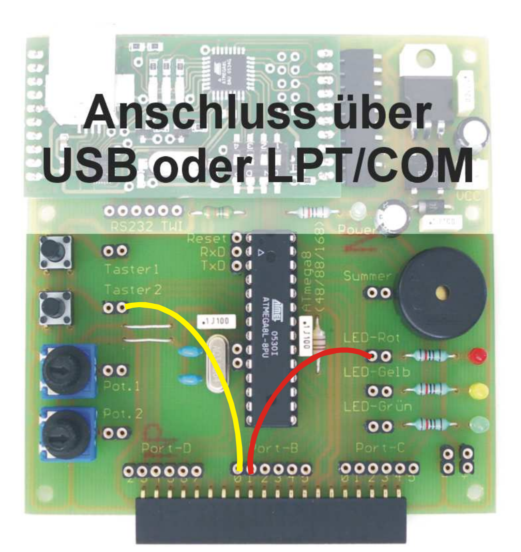


Abbildung 26

Nachdem das Programm erfolgreich übersetzt und auf den Controller übertragen wurde, kann die Anwendung getestet werden. Dazu ist das myAVR-Controlcenter zu öffnen und die vorgegebenen Verbindungen auf dem Board zu stecken. Über die Schaltfläche „Start/Stopp“ kann das myAVR Board ein und ausgeschaltet werden.

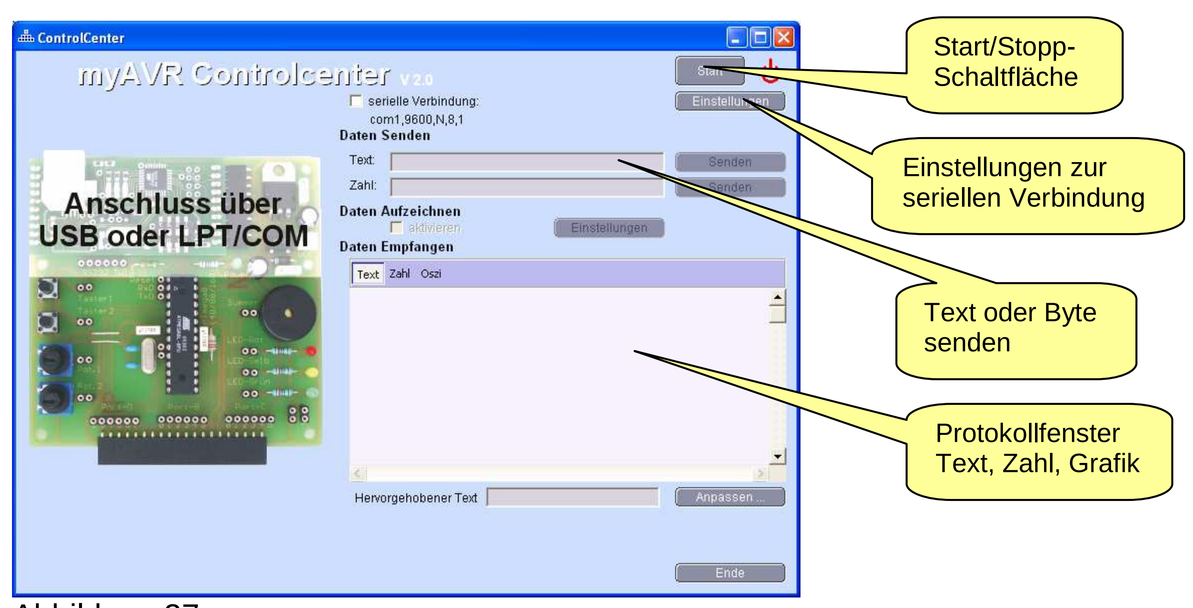


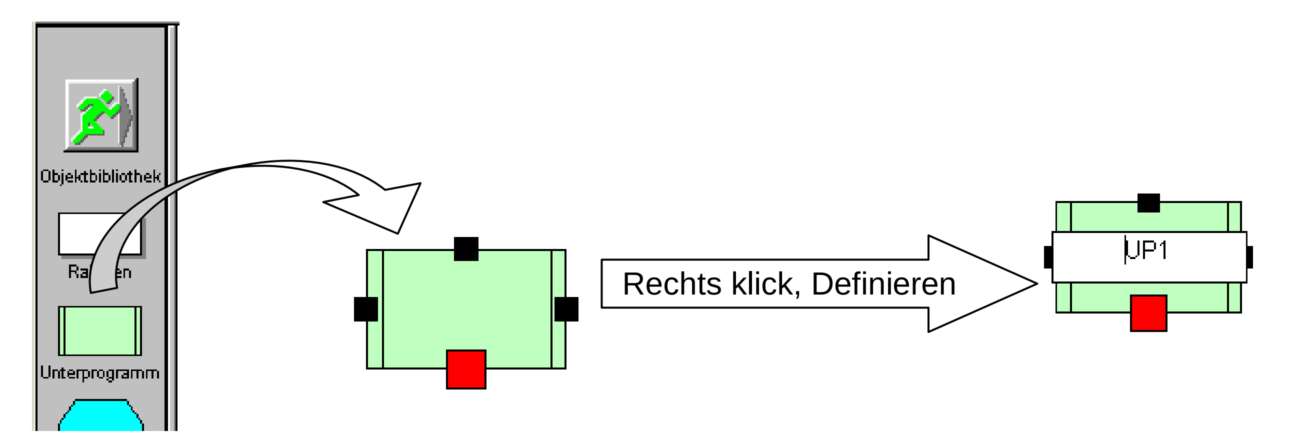
Abbildung 27

9.8 Unterprogrammtechnik im PAP

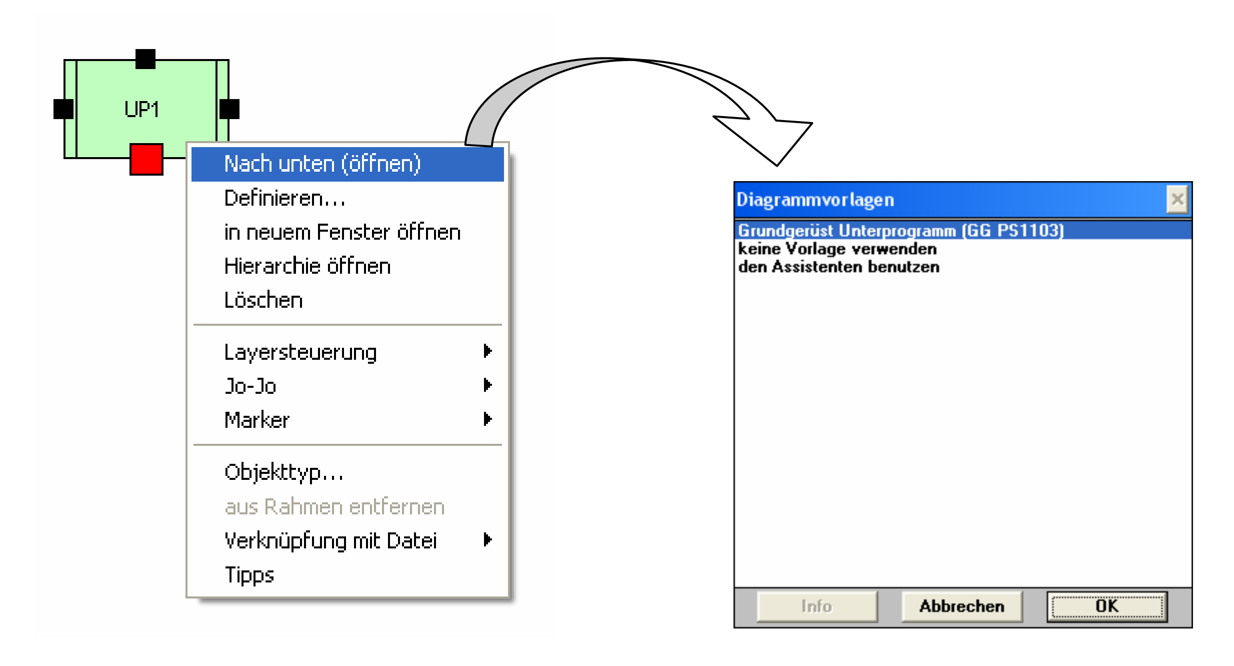
Unterprogramme sind ein wichtiges Gestaltungsmittel für übersichtliche Mikrocontrollerprogramme. Sie werden für in sich abgeschlossene Aufgaben (Verarbeitungsschritte) benutzt, die auch mehrfach im Gesamtprogramm genutzt werden können.

9.8.1 Anlegen eines Unterprogramms

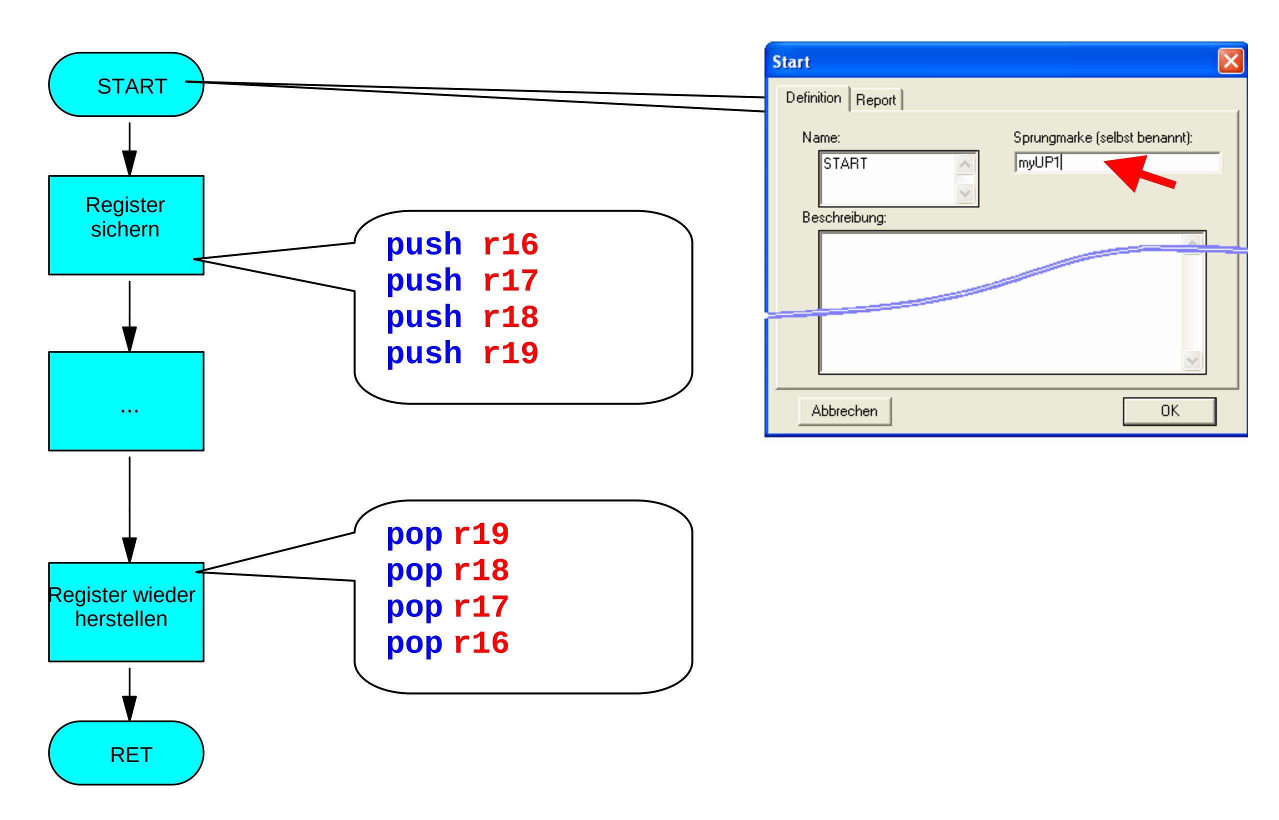
Ziehen Sie den Objekttyp „Unterprogramm“ aus der Objektbibliothek in das gewünschte Diagramm. Mit Doppelklick oder über rechte Maustaste -> Kontextmenü -> Definieren auf dem Element können Sie dem Unterprogramm einen Namen geben (Beachte: Menüfolge Einstellungen/Menü bei Doppelklick).



Damit ist ein Objekt angelegt, welches im aktuellen Diagramm als Aufruf (call) des Unterprogramms zu verstehen ist. Die Funktionalität des Unterprogramms wird in einem gesonderten Programmablaufplan für das Unterprogramm entworfen. Dazu ist das Diagramm „unter" bzw. „hinter" Unterprogramm zu öffnen. Um das zum Unterprogramm zugehörige Diagramm zu öffnen, wählen Sie auf dem Objekt rechte Maustaste -> Kontextmenü -> Nach unten (öffnen).

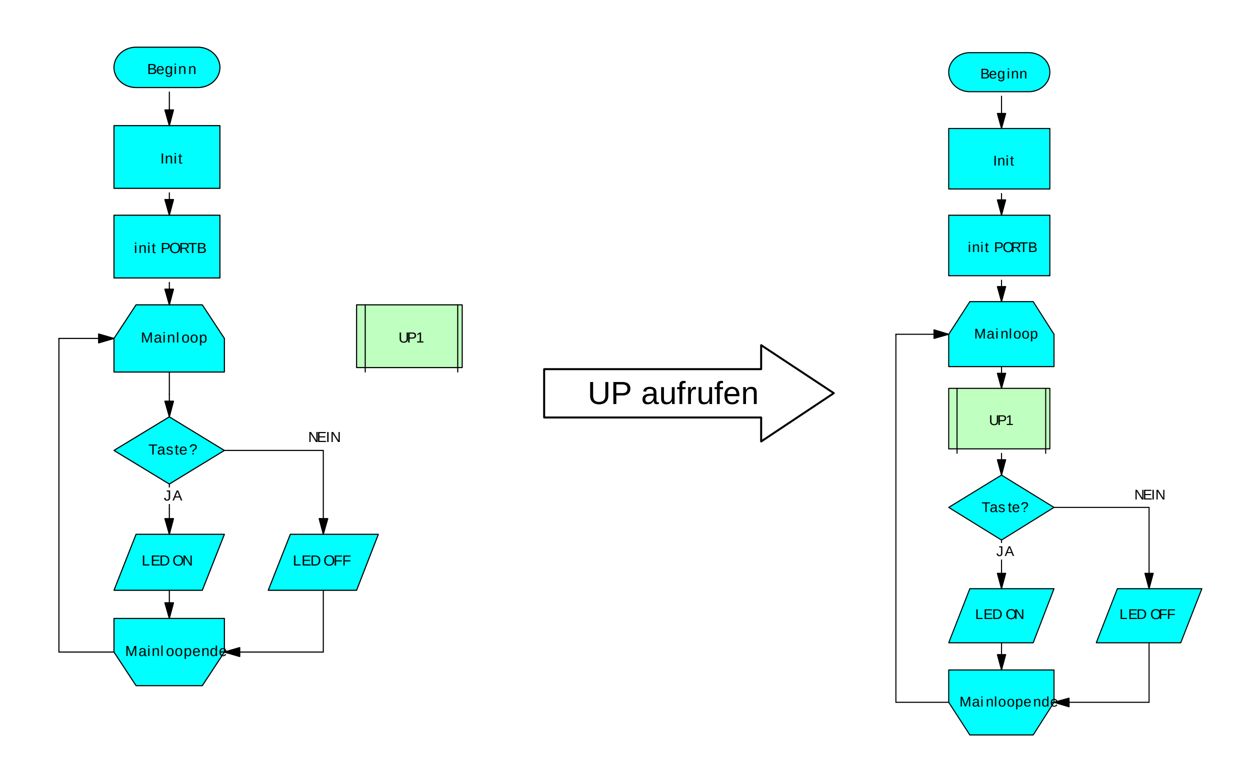


Sie erhalten eine Vorlagenliste für die Grundstruktur von Unterprogrammen. Bitte laden Sie die Vorlage „Grundgerüst Unterprogramm". Auf dem Objekt „START" können Sie eine benutzerdefinierte Sprungmarke festlegen (Rechtsklick -> Definieren), die durch den Codegenerator erstellt und verwendet werden soll. Die Vorlage muss entsprechend der vorgesehen Logik abgeändert werden.



9.8.2 Ein Unterprogramm aufrufen

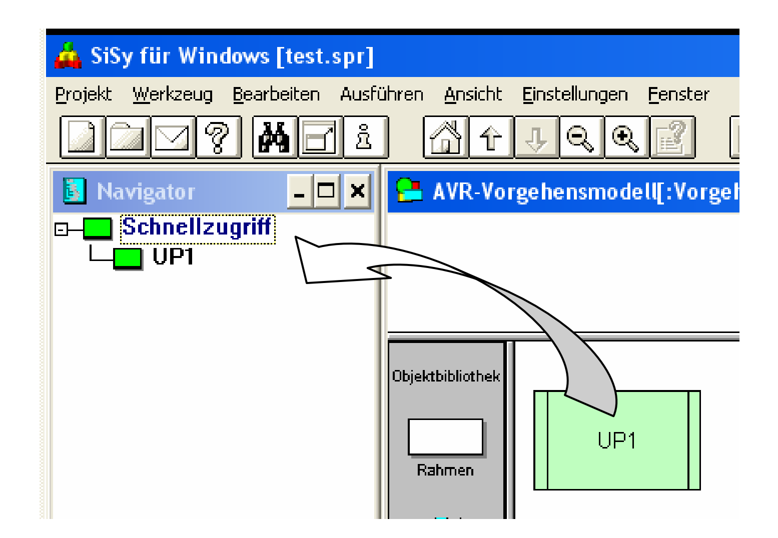
Das Unterprogrammsymbol muss zum Aufruf an der entsprechenden Position im Programmablaufplan eingefügt werden. Der Codegenerator erzeugt dann entsprechend einen Unterprogrammaufruf und den Code für das Unterprogramm selbst. Dazu ist in das Diagramm zurückzukehren, in dem das Symbol „Unterprogramm" angelegt wurde (zum Beispiel rechte Maustaste -> Kontextmenü-> nach oben…). Das Unterprogramm ist korrekt eingebunden, wenn es vollständig und eindeutig im Programmfluss integriert ist (mindestens ein eingehender Pfeil und genau ein ausgehender Pfeil).



9.8.3 Unterprogramme mehrmals benutzen

Ein wesentliches Merkmal von Unterprogrammen ist, dass diese von verschiedenen Stellen im Programm aufgerufen (call) werden können und auch dorthin zurückkehren (return). Um diese Möglichkeit zu nutzen, bietet SiSy das Anlegen von Referenzen an. Vergleichen Sie dazu Absatz 3.2 „Die Modellierungselemente von SiSy“. Um ein Unterprogramm zu referenzieren (wiederholend zeigen und einbinden) gehen Sie wie folgt vor:

1. zeigen Sie im Navigator das gewünschte Unterprogramm an
   a. über den Schnellzugriff,
      dort lässt sich das Original per Drag & Drop ablegen, oder
   b. über die Navigatorsortierung „Unterprogramme“
      Navigator -> rechte Maustaste -> Kontextmenü -> Unterprogramme
2. das gewünschte Zieldiagramm, in dem das Unterprogramm verwendet werden soll, öffnen
3. per Drag und Drop das Unterprogramm in das Zieldiagramm ziehen, dabei wird nur eine Referenz (Link) auf das Original angelegt.
4. Referenz wie oben beschrieben in den Programmfluss integrieren



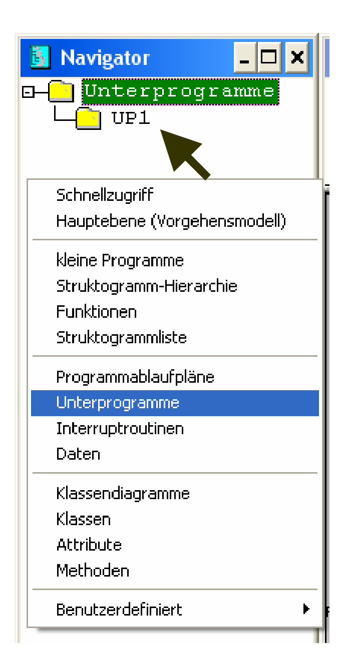


Abbildung 29: Unterprogramme im Navigator anzeigen

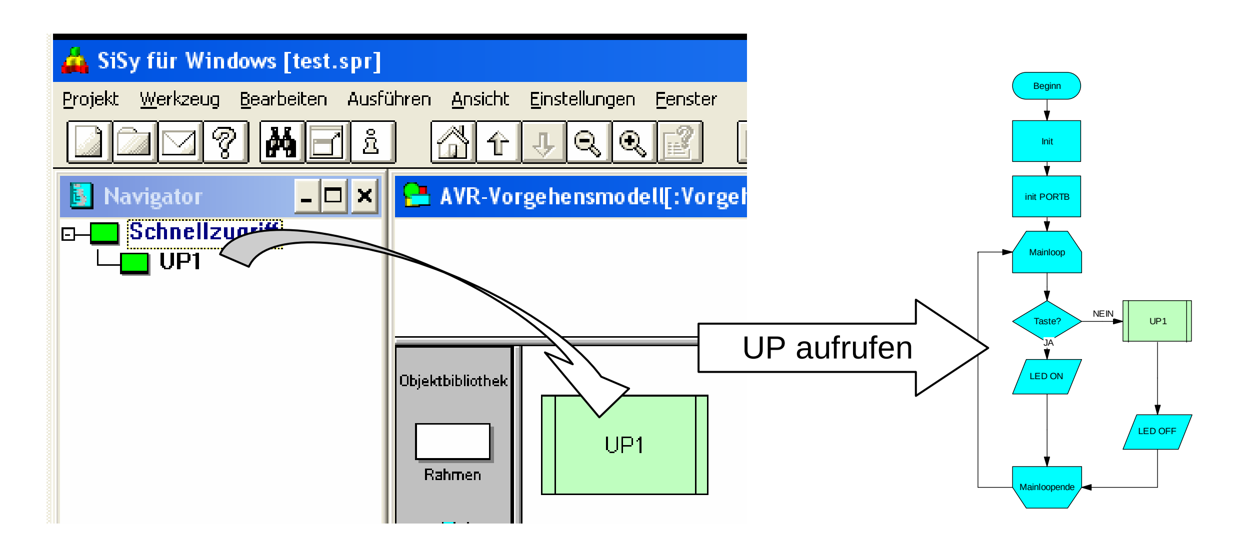
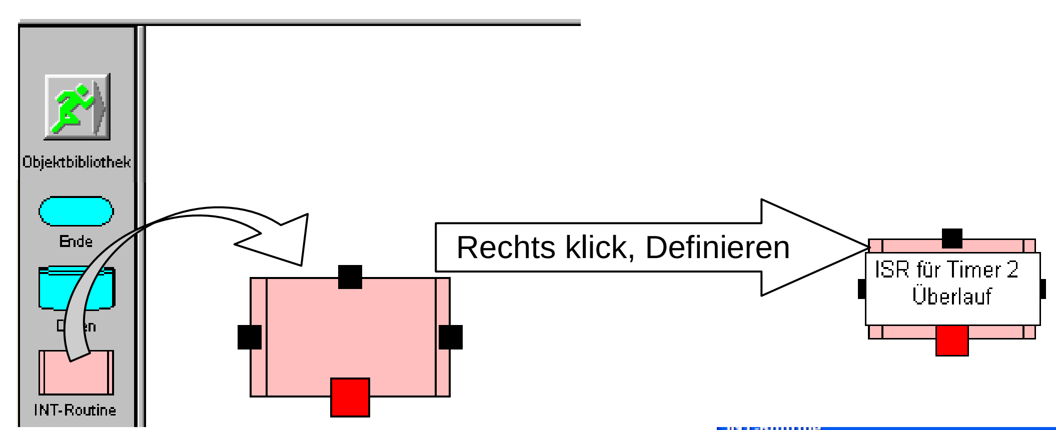
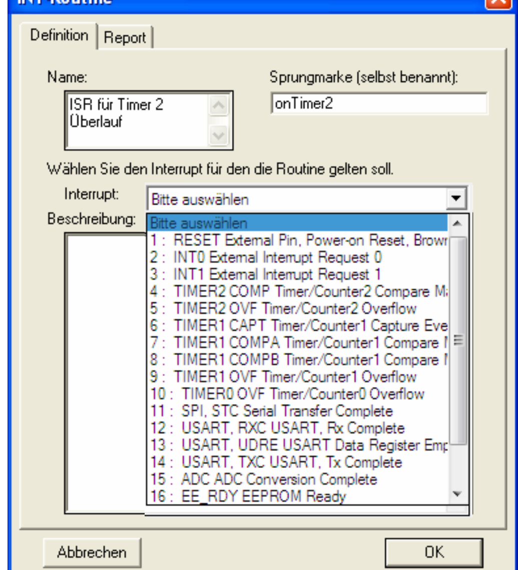


Abbildung 28: Unterprogramm referenzieren

9.9 Interrupt-Service-Routinen (ISR) im PAP

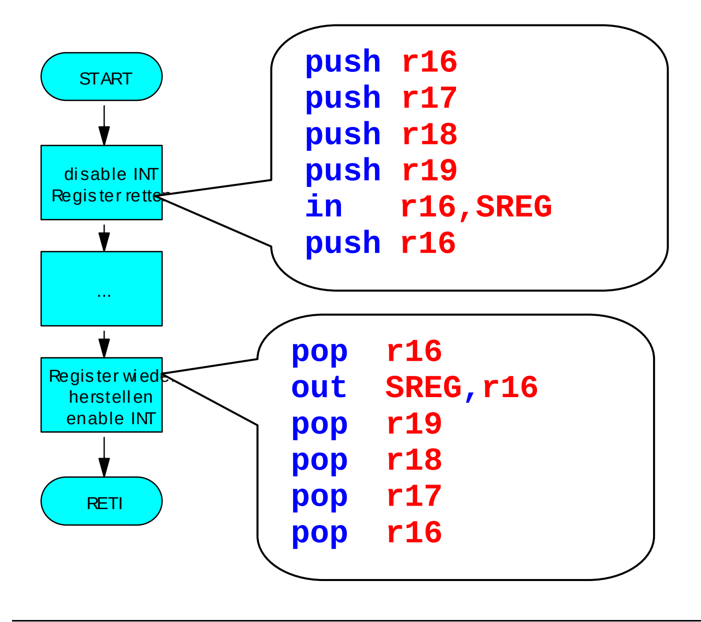
Interrupt-Service-Routinen (im weiteren ISR) sind besondere Formen von Unterprogrammen. Diese werden von einer Interruptquelle des Mikrocontrollers (Timer, ADC, UART, usw.) bei entsprechenden Ereignissen automatisch an beliebiger Stelle im Programmfluss aufgerufen (Unterbrechung, engl. Interrupt). Es ist demzufolge nicht vorgesehen, eine ISR in den Programmfluss zu integrieren.
Um eine ISR zu erzeugen, ziehen Sie ein Objekt vom Typ INT-Routine aus der Objektbibliothek in das Diagramm des Hauptprogramms und definieren einen Namen.

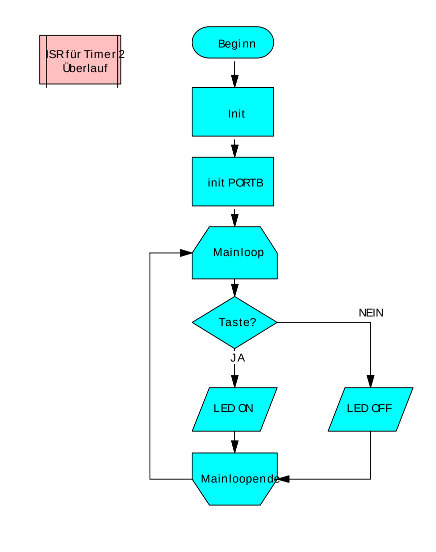




Über rechte Maustaste -> Kontextmenü -> Definieren … legen Sie die Sprungmarke und den Typ des Interrupt fest. Damit erfolgt durch den Codegenerator die Zuordnung der ISR zum entsprechenden Interruptvektor. Beachten Sie, dass die Liste der Interrupts abhängig ist vom gewählten Controllertyp.

Zum Entwerfen der ISR-Logik wählen Sie auf dem Objekt rechte Maustaste -> Kontextmenü -> nach unten … Ihnen wird das Grundgerüst einer ISR als Diagrammvorlage angeboten. Laden Sie die Diagrammvorlage.
Vervollständigen Sie danach die ISR-Logik. Die ISR wird nicht in den Programmfluss integriert.



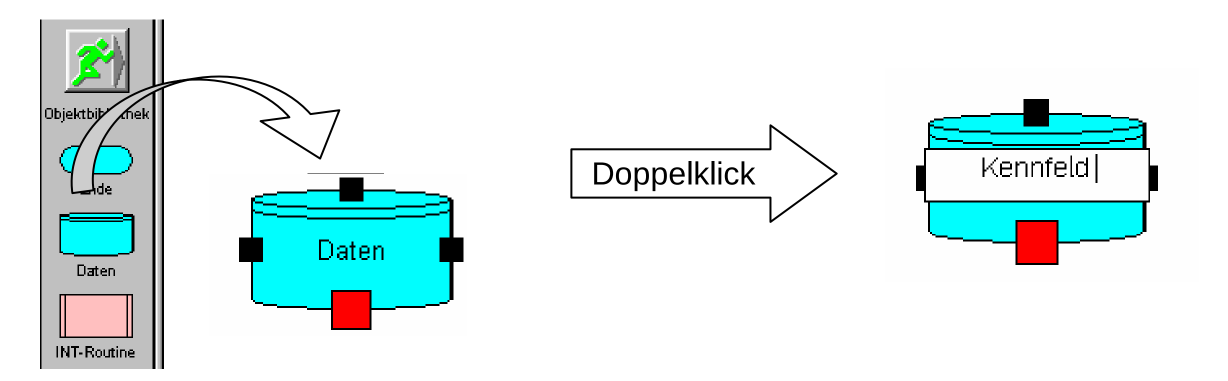


9.10 Daten im PAP

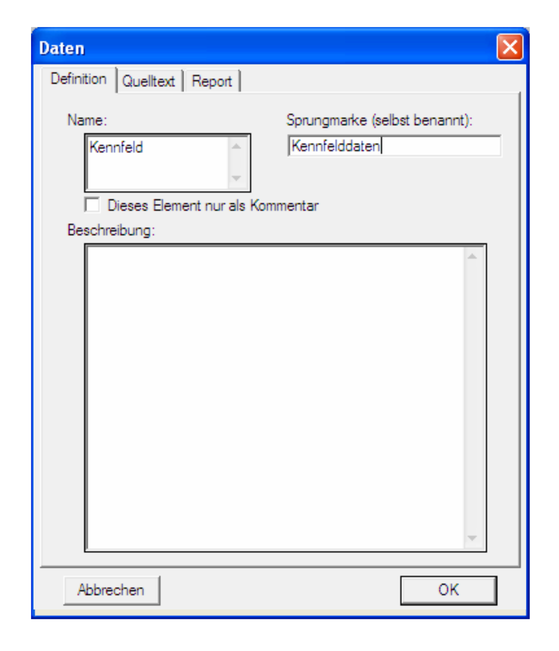
Konstante Daten, die im Programmspeicher (FLASH) des Mikrocontrollers abgelegt werden, können als Datenobjekt deklariert werden. Die im Datenobjekt deklarierten Daten werden durch den Codegenerator immer ans Ende des Quellcodes gestellt. Die zu generierende Marke/Marken für die Datenelemente können vom Entwickler frei gewählt werden.

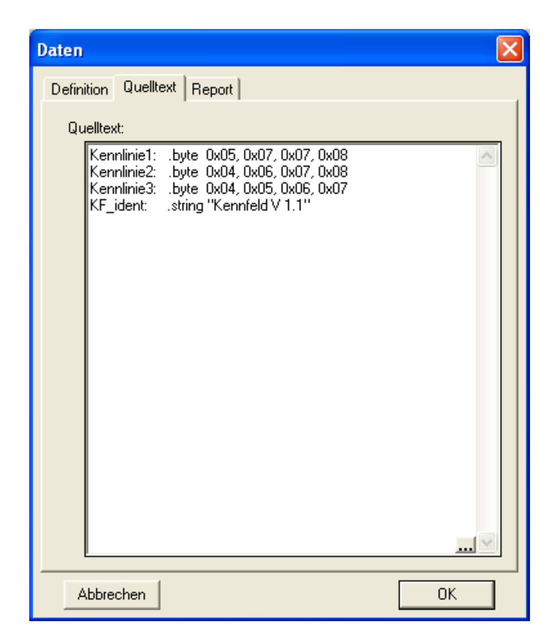
9.10.1 Anlegen eines Datenobjektes

Zum Anlegen eines Datenobjektes ziehen Sie das betreffende Objekt per Drag & Drop aus der Objektbibliothek in das gewünschte Diagramm. Per Doppelklick können Sie einen Namen vergeben (Beachte: Menüfolge *Einstellungen/Menü bei Doppelklick*).



Die Marke und die Daten selbst können über rechte Maustaste -> Kontextmenü -> Definieren… festgelegt werden.





9.10.2 Datenobjekt benutzen

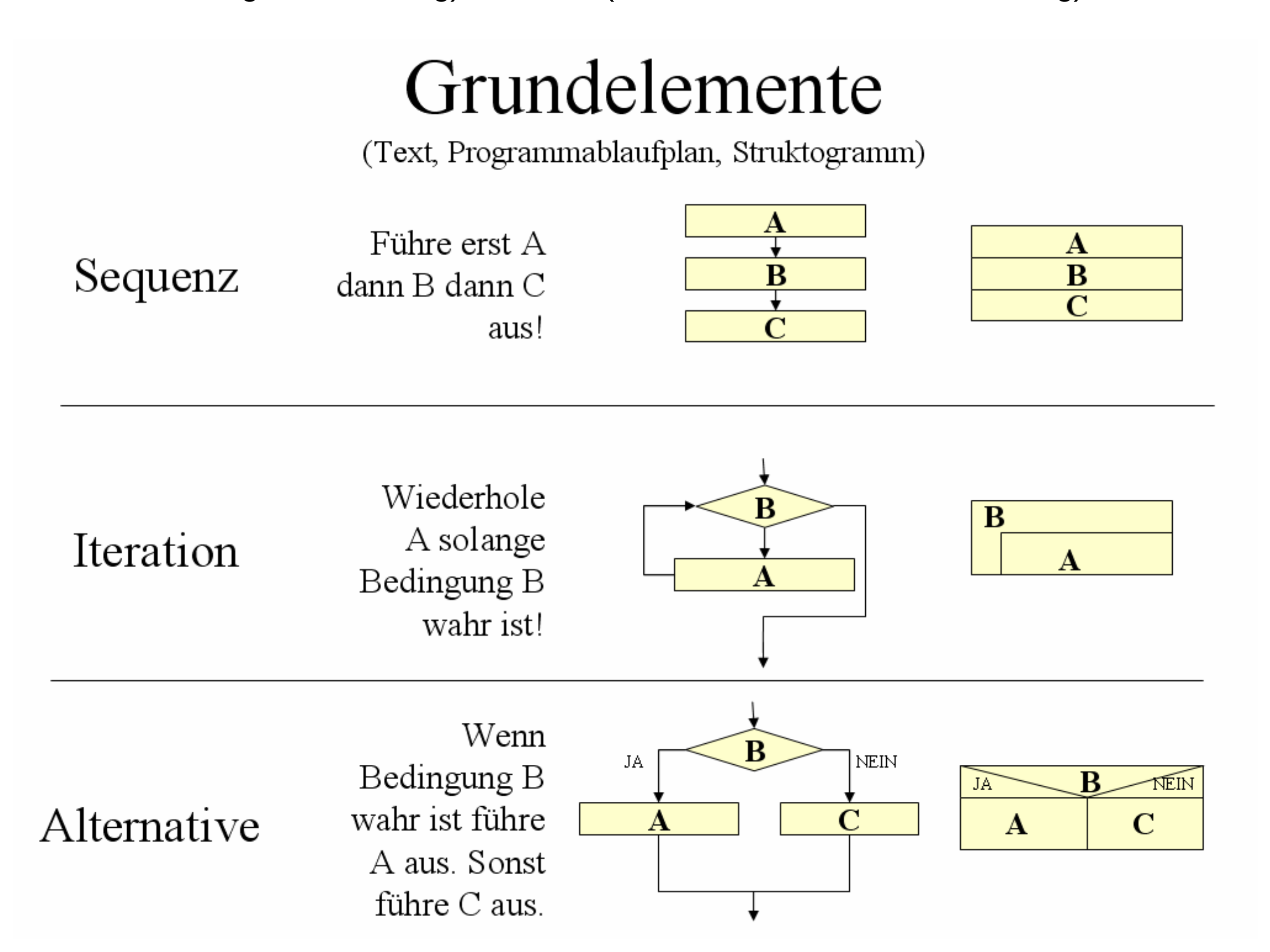
Im Quellcode werden die Daten über die vergebenen Markennamen angesprochen.

```
DEV Programmablaufplan[:testPAP]
{
0001    ; Kennfelddaten in Adressgerister Laden
0002    ldi r30,lo8(Kennlinie1)
0003    ldi r31,hi8(Kennlinie1)
0004    ...
0005
0006
}
```


10 Entwicklung eines Struktogramms mit SiSy AVR

10.1 Einleitung

Struktogramme (SG) oder auch Nassi-Shneiderman-Diagramme sind ein Entwurfsmittel der strukturierten Programmierung. Strukturiert meint in dem Fall, dass zur Realisierung eines Algorithmus auf das Verwenden von Sprunganweisungen (Goto, Jump) verzichtet wird. Für das Formulieren eines Algorithmus stehen dem Entwickler drei normierte Grundstrukturen zur Verfügung: Sequenz (Folge von Anweisungen), Alternative (Auswahl bzw. bedingte Anweisung), Iteration (Schleife, wiederholte Anweisung).



Struktogramme werden als Darstellungsmittel für strukturierte Sprachen wie C oder PASCAL verwendet, da hier im Gegensatz zu Assembler in der Regel auf Sprunganweisungen verzichtet wird.

10.2 Aufgabenstellung

Um die Arbeitsweise und den Umgang mit dem Struktogrammeditor in SiSy zu erlernen, realisieren Sie bitte das folgende Beispiel.

Es soll eine Mikrocontrollerlösung entwickelt werden, bei der auf Tastendruck eine LED eingeschaltet wird. Dabei soll der Taster an Port B 0 und die LED an Port B 1 angeschlossen werden

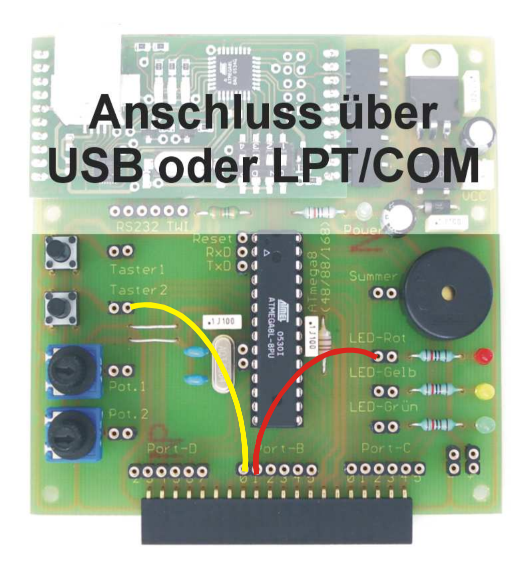
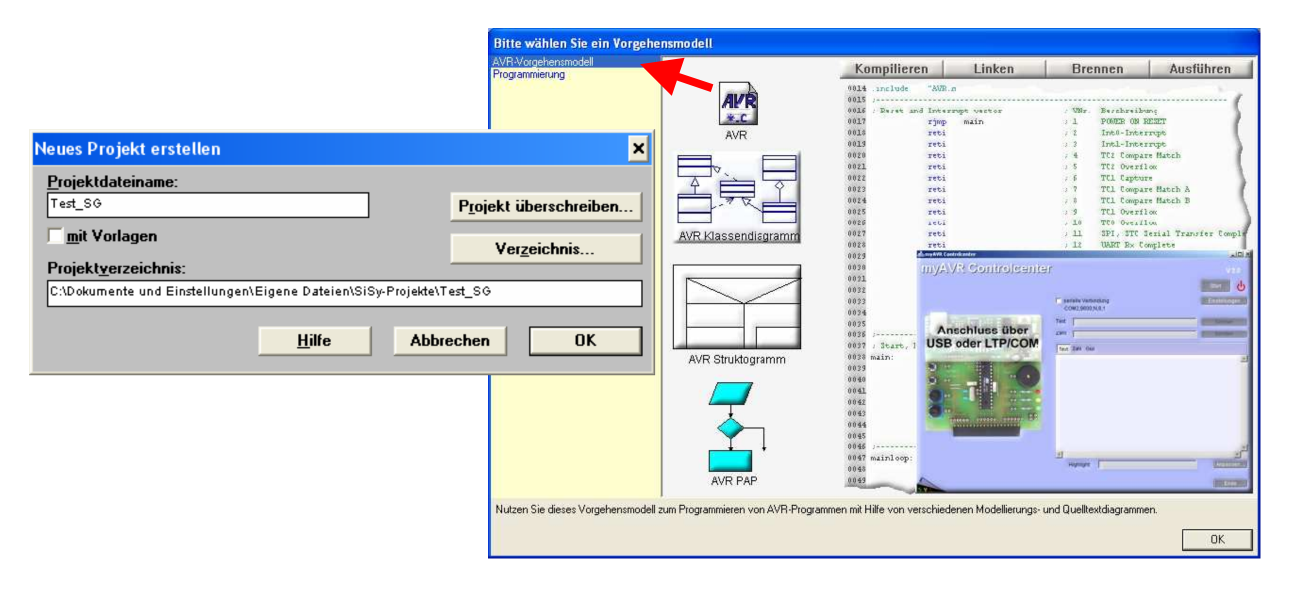


Abbildung 30: Beispiel für SG

10.3 Vorbereitung

Das Struktogramm kann je SiSy Projekt ein Hauptprogramm verwalten. Es ist also nötig, für jedes neue Struktogramm-Projekt auch ein neues SiSy-Projekt zu erzeugen. Starten Sie ggf. SiSy.

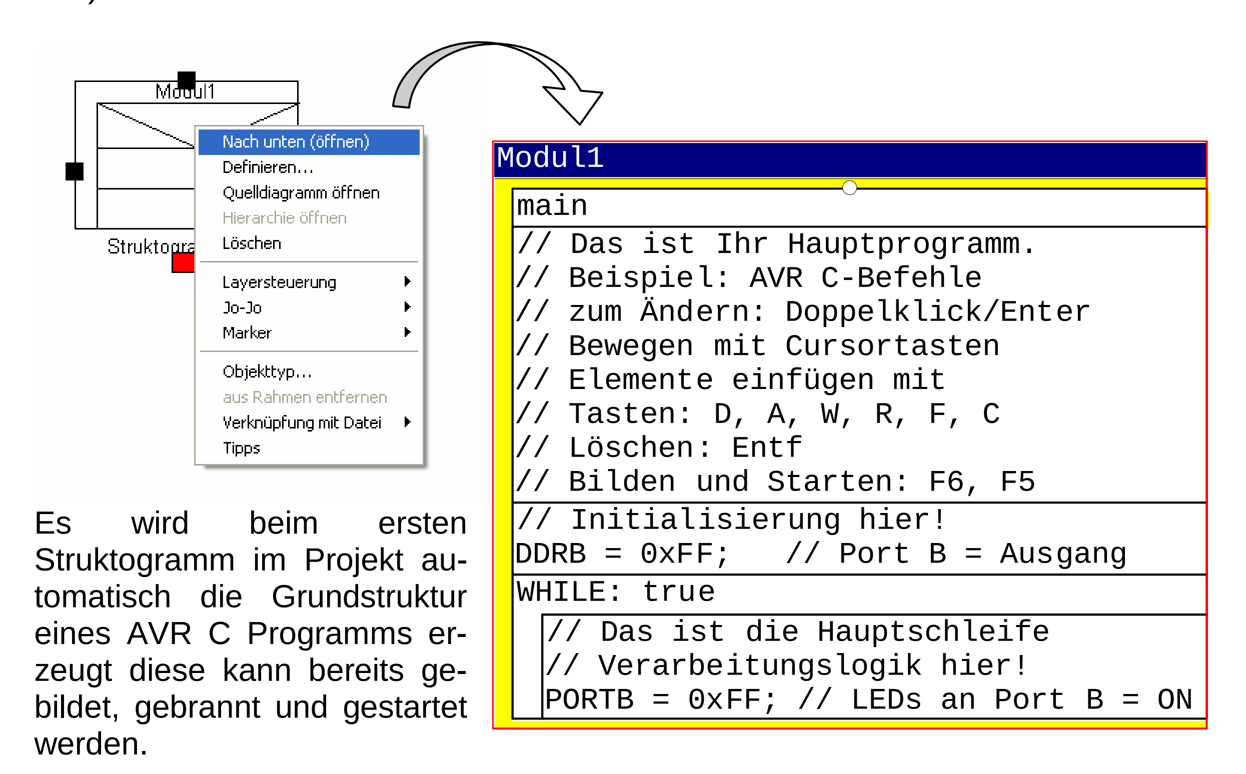
Erstellen Sie ein neues SiSy-Projekt (Hauptmenu -> Projekt -> Neu) mit dem Namen „Test_SG". Wählen Sie das Vorgehensmodell „AVR-Vorgehensmodell". Danach erscheint ein Dialogfenster mit verschiedenen Diagrammvorlagen.



Legen Sie im weiteren Verlauf die AVR - Grundeinstellungen für das Projekt fest. Ziehen Sie ein Objekt vom Typ SG (Struktogramm) per Drag & Drop in das Diagrammfenster. Legen Sie den Namen für das Struktogramm fest.

10.4 Struktogramm entwickeln

Um das Struktogramm zu entwickeln, muss das Struktogrammfenster geöffnet werden. Wählen Sie dazu auf dem Objekt rechte Maustaste -> Kontextmenü -> nach unten (öffnen).



Es wird beim ersten Struktogramm im Projekt automatisch die Grundstruktur eines AVR C Programms erzeugt diese kann bereits gebildet, gebrannt und gestartet werden.

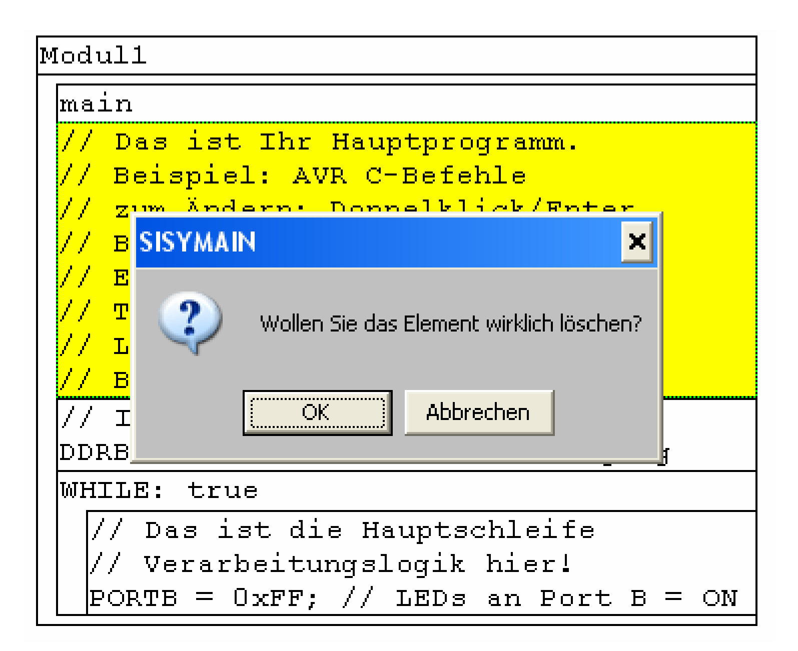
Es empfiehlt sich im Struktogramm mit der Tastatur zu arbeiten.

Tastenbelegung (Auszug) im Struktogramm:

F3	öffnet den Quelltext
F5	startet das Programm (bei SiSy AVR das myAVR Controlcenter)
F8	Codegenerierung des aktuellen Struktogramms
F9	übersetzt das Struktogramm
ESC	Ebene nach oben
Num `+`	selektiertes Objekt wird vergrößert
Num `-`	selektiertes Objekt wird verkleinert
Enter	ermöglicht die Bearbeitung des Quelltextes im Struktogramm-Editor
Entf	löschen
T	ermöglicht die Bearbeitung des Titels
K	ermöglicht die Bearbeitung des Kommentars
B	Begin neue Funktion einfügen
S	Sequenz einfügen (Folge/Block)
D	Do einfügen (elementare Aktion)
W	While-Schleife einfügen (kopfgesteuerte Schleife)
R	Repeat-Schleife einfügen (fußgesteuerte Schleife)
F	For-Schleife einfügen (Zählschleife)
A	Alternative einfügen
C	Case einfügen (Fallunterscheidung)
I	If-Zweig einfügen
E	Exit einfügen
L	Loop einfügen
Cursortasten	Bewegung im Struktogramm

Führen Sie folgende Arbeitsschritte aus, um den Algorithmus für die oben genannte Aufgabe zu entwerfen:

1. Löschen Sie den Kommentar, selektieren Sie dazu das Element zum Beispiel mit der Maus oder den Cursortasten. Betätigen Sie dann die Taste „Entf“ und bestätigen Sie die Sicherheitsabfrage mit „OK“.



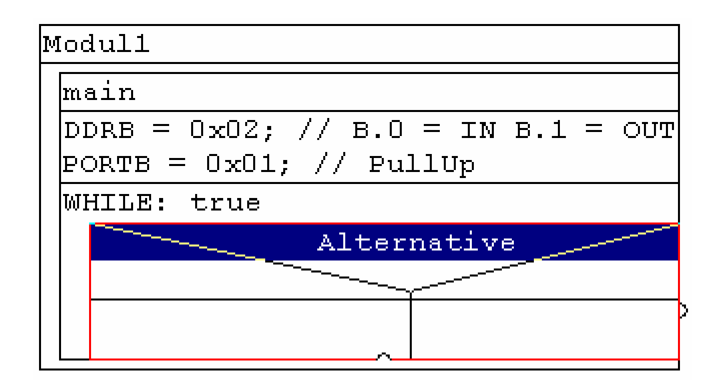
2. Konfigurieren Sie Port B 0 als Eingang und Port B 1 als Ausgang. Selektieren Sie dazu die Initialisierungssequenz. Mit ENTER kommen Sie in den Editiermodus. Mit ESC verlassen Sie den Editiermodus.

```
Modul1
main
DDRB = 0x02; // B.0 = IN B.1 = OUT
PORTB = 0x01; // PullUp
WHILE: true
// Das ist die Hauptschleife
// Verarbeitungslogik hier!
PORTB = 0xFF;
```

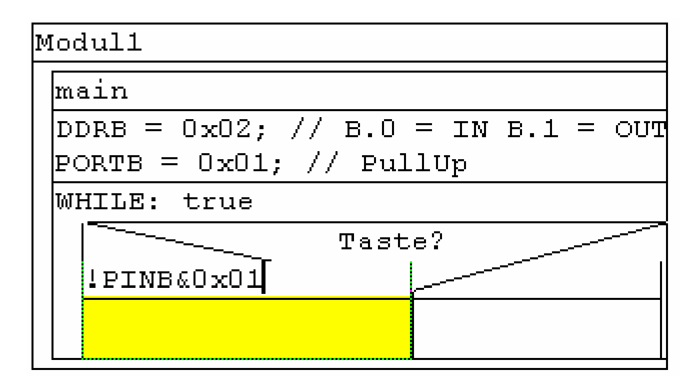
3. Löschen Sie das DO-Element in der Mainloop (WHILE-Schleife). Dazu selektieren Sie mit der Maus oder den Cursortasten das DO-Element und betätigen die Taste „Entf“.

```
Modul1
main
DDRB = 0x02; // B.0 = IN B.1 = OUT
PORTB = 0x01; // PullUp
WHILE: true
```

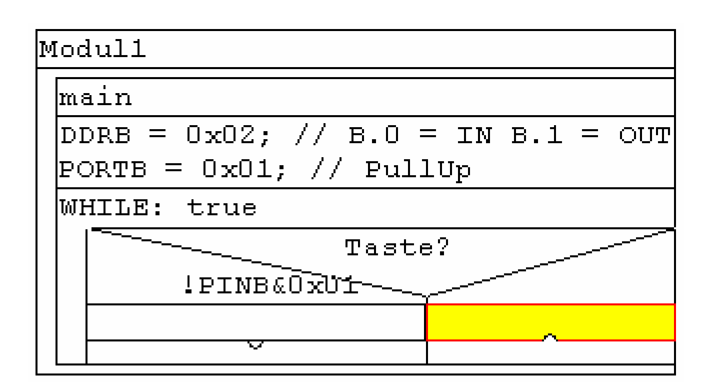
4. Ergänzen Sie die Mainloop durch eine Alternative. Dazu betätigen Sie die Taste „A“. Achten Sie darauf, dass die Alternative innerhalb der Mainloop liegt.



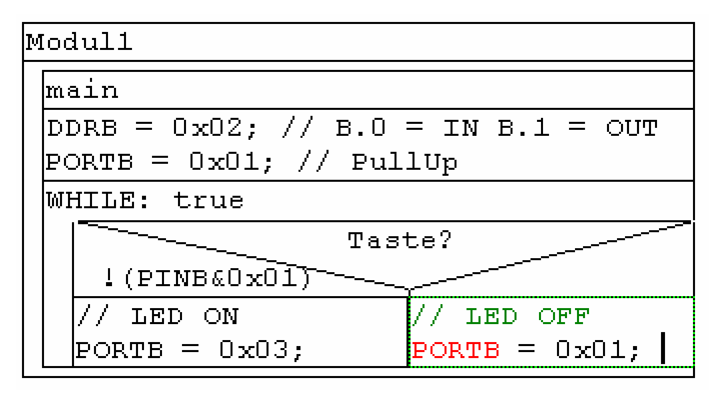
5. Geben Sie den Titel und die Bedingung für die Alternative ein. Dazu selektieren Sie das gewünschte Element und betätigen die Taste ENTER oder Doppelklick mit der Maus.

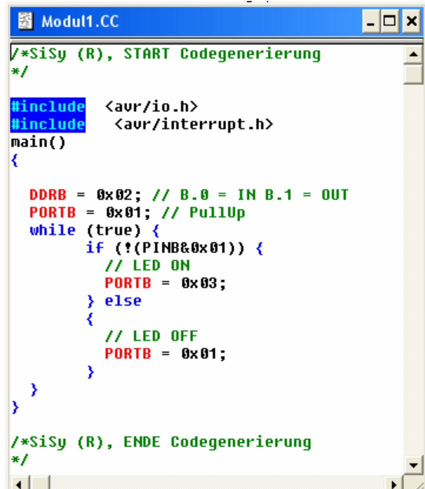


6. als nächstes sind die Aktionen (DO-Elemente) in der Alternative zu ergänzen. Selektieren Sie dazu den Alternativzweig mit der Maus oder den Cursortasten. Betätigen Sie dann jeweils die Taste „D“.



7. Tragen Sie den C-Quellcode zum Ein- und Ausschalten der LED in die DO-Elemente ein. Selektieren Sie dazu das betreffende DO-Element und wählen ENTER oder Doppelklick. Mit ESC verlassen Sie den Editiermodus.





```
/*SiSy (R), START Codegenerierung
*/
#include  <avr/io.h>
#include  <avr/interrupt.h>
main()
{
  DDRB = 0x02; // B.0 = IN B.1 = OUT
  PORTB = 0x01; // PullUp
  while (true) {
        if (!(PINB&0x01)) {
          // LED ON
          PORTB = 0x03;
        } else
        {
          // LED OFF
          PORTB = 0x01;
        }
  }
}
/*SiSy (R), ENDE Codegenerierung
*/
```

8. Generieren Sie den Quellcode für dieses Struktogramm. Betätigen Sie die Taste F8 und danach die Taste F3.

10.5 Übersetzen, Brennen und Testen

Der so generierte Quellcode kann jetzt kompiliert, gelinkt und gebrannt werden. Wenn die HEX-Datei erfolgreich übertragen wurde, kann zum Testen das myAVR Controlcenter gestartet werden. Nutzen Sie dazu die Toolbox „Objekt“.

Alles Bilden und Brennen

START myAVR Controlcenter

```
Compiling...
Generiere Linkerdatei....
Linke Datei <>
Linking...
Bilden und Brennen des Hauptprogramms  Modul1
Compiliere Modul1.cc
keine Compilerfehler!
Linke Modul1.cc
keine Linkerfehler!
Brenne Modul1.hex
Die Datei 'Modul1.hex' wurde erfolgreich gebrannt.
Modul1.hex uebertragen!
Starten des Mikrocontrollers mit Taste F5 ...
```

myAVR Controlcenter

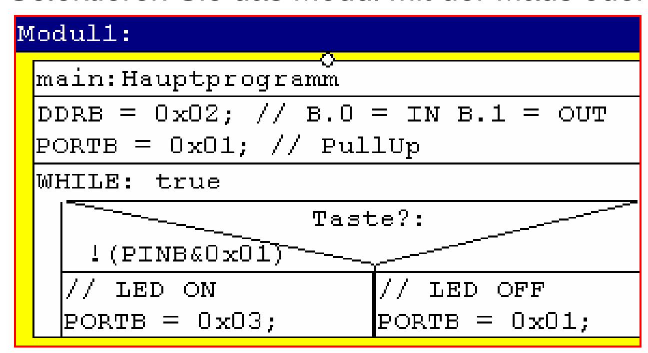
Anschluss über USB oder LPT/COM

START des myAVR Boards

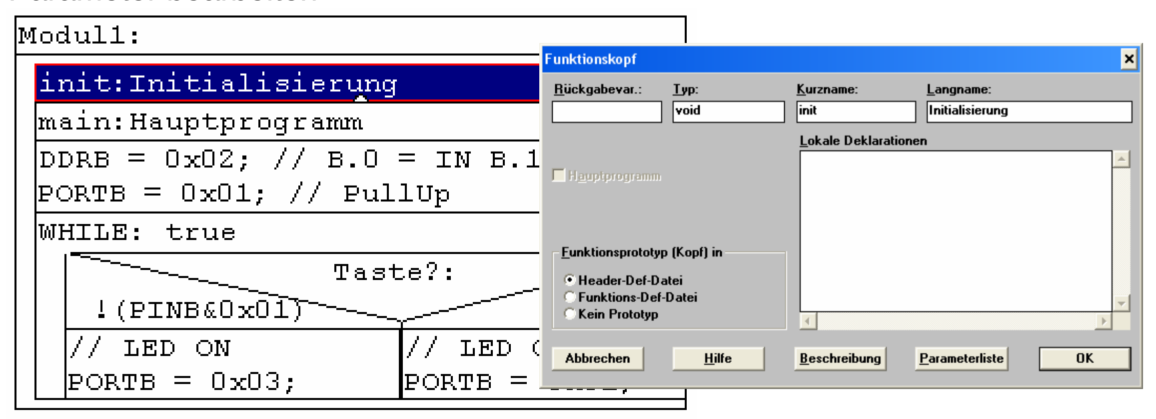
10.6 Funktionen (Unterprogramme) im Struktogramm

Funktionen sind in der Sprache C das wichtigste Instrument zur übersichtlichen und wartungsfreundlichen Gestaltung von Programmen. Es gibt Standardfunktionen wie **`itoa`** oder **`sprintf`** die durch die Verwendung von Bibliotheken wichtige, immer wieder benötigte Algorithmen zur Verfügung stellen. Für den C-Programmierer ist die Entwicklung eigener Funktionen genauso wichtig wie die Verwendung von Bibliotheksfunktionen. Um eine Funktion im Struktogramm zu entwerfen, gehen Sie wie folgt vor (Erweiterung des Beispiels durch die Funktion **`void init()`**, Auslagern der Initialisierung):

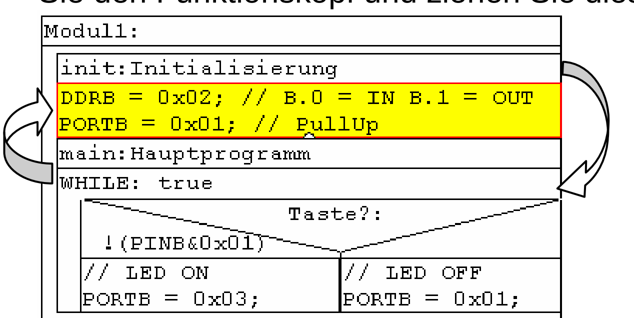
1. Selektieren Sie das Modul mit der Maus oder den Cursortasten.

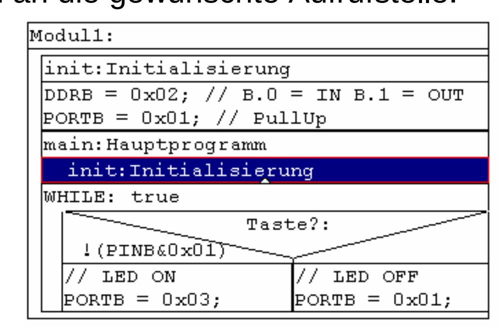


2. Fügen Sie durch Betätigen der Taste „B“ (Begin / Funktion) eine neue Funktion ein. Mit ENTER oder Doppelklick auf der Funktion können Sie Name, Typ und Parameter bearbeiten.



3. Verschieben Sie per Drag & Drop die Aktion „Initialisierung“ in die Funktion **`init`**. Den Funktionsaufruf erzeugen Sie ebenfalls per Drag & Drop. Selektieren Sie den Funktionskopf und ziehen Sie diesen an die gewünschte Aufrufstelle.



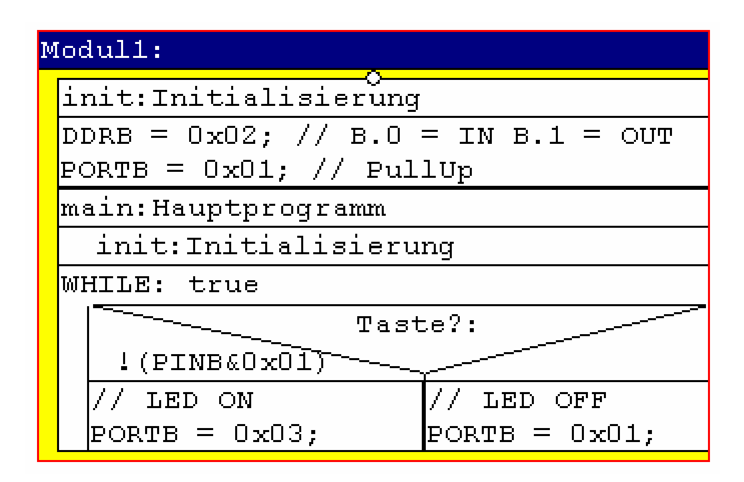


Mit der Tastenfolge F8 und F3 können Sie sich das Ergebnis der Umstrukturierung im Quellcode ansehen.

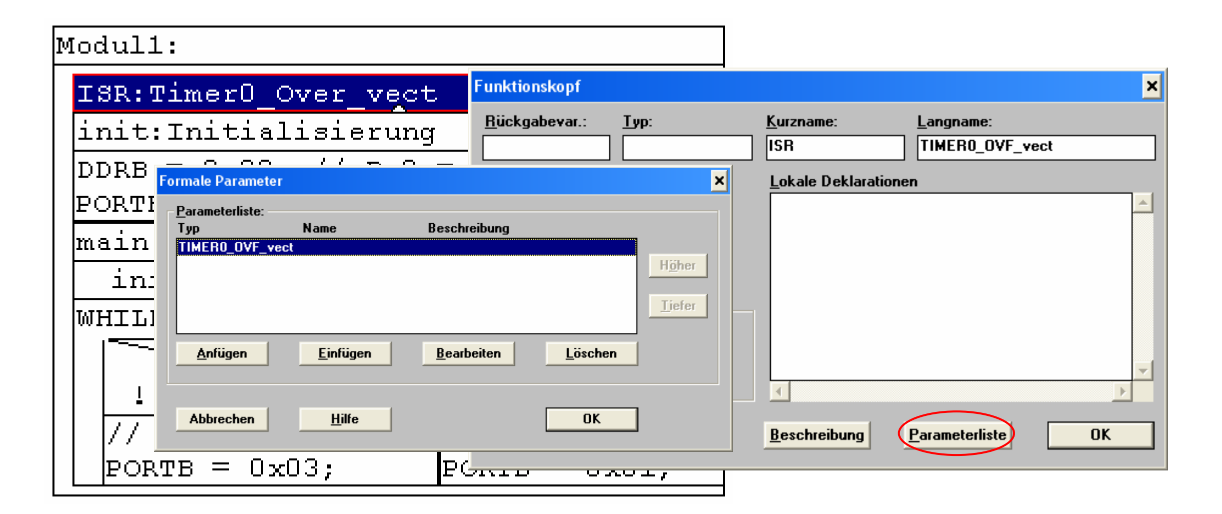
10.7 Interrupt-Service-Routinen (ISR) im Struktogramm

Interrupt-Service-Routinen (im Weiteren ISR) werden in AVR C durch das Schlüsselwort ISR gekennzeichnet. Sie bilden eigenständige Funktionen. Die Besonderheit bei der Realisierung einer ISR liegt darin, dass es sich hier um ein C-Makro handelt und nicht um eine echte Funktion. Um eine ISR im Struktogramm zu entwerfen, gehen Sie wie folgt vor (Erweiterung des Beispiels um die **`ISR Timer0 Overflow`**):

1. Selektieren Sie das Modul mit der Maus oder den Cursortasten.



2. Fügen Sie mit der Taste „B“ eine neue Funktion ein. Per Doppelklick oder Taste ENTER können Sie Name, Typ und Parameter definieren (Typ = Leerzeichen, Kurzname = ISR, Parametertyp = TIMER0_OVF_vect, Parametername = Leerzeichen).



3. Fügen Sie entsprechende Elemente (Do, Alternative, While usw.) für die ISR-Logik ein.

```
Modul1:
ISR:TIMER0_OVF_vect
//hier die ISR-Logik entwickeln
```

11 Entwicklung eines Klassendiagramms mit SiSy AVR

11.1 Einleitung

Mit objektorientierten Programmiersprachen hat der Entwickler mächtige Sprachmittel, um komplexe Systeme realisieren zu können. C++ ist eine weit verbreitete objektorientierte Standardsprache. Als Visualisierungsmittel objektorientierter Programme gilt die international standardisierte Beschreibungssprache UML (Unified Modeling Language). SiSy AVR bietet dem Entwickler das UML Klassendiagramm mit Codegenerierung für AVR C++. Der folgende Abschnitt beschreibt die Handhabung des Klassendiagramms in SiSy AVR. Die Abbildung 31 zeigt Ihnen eine Kurzübersicht der Modellierungselemente des UML Klassendiagramms.

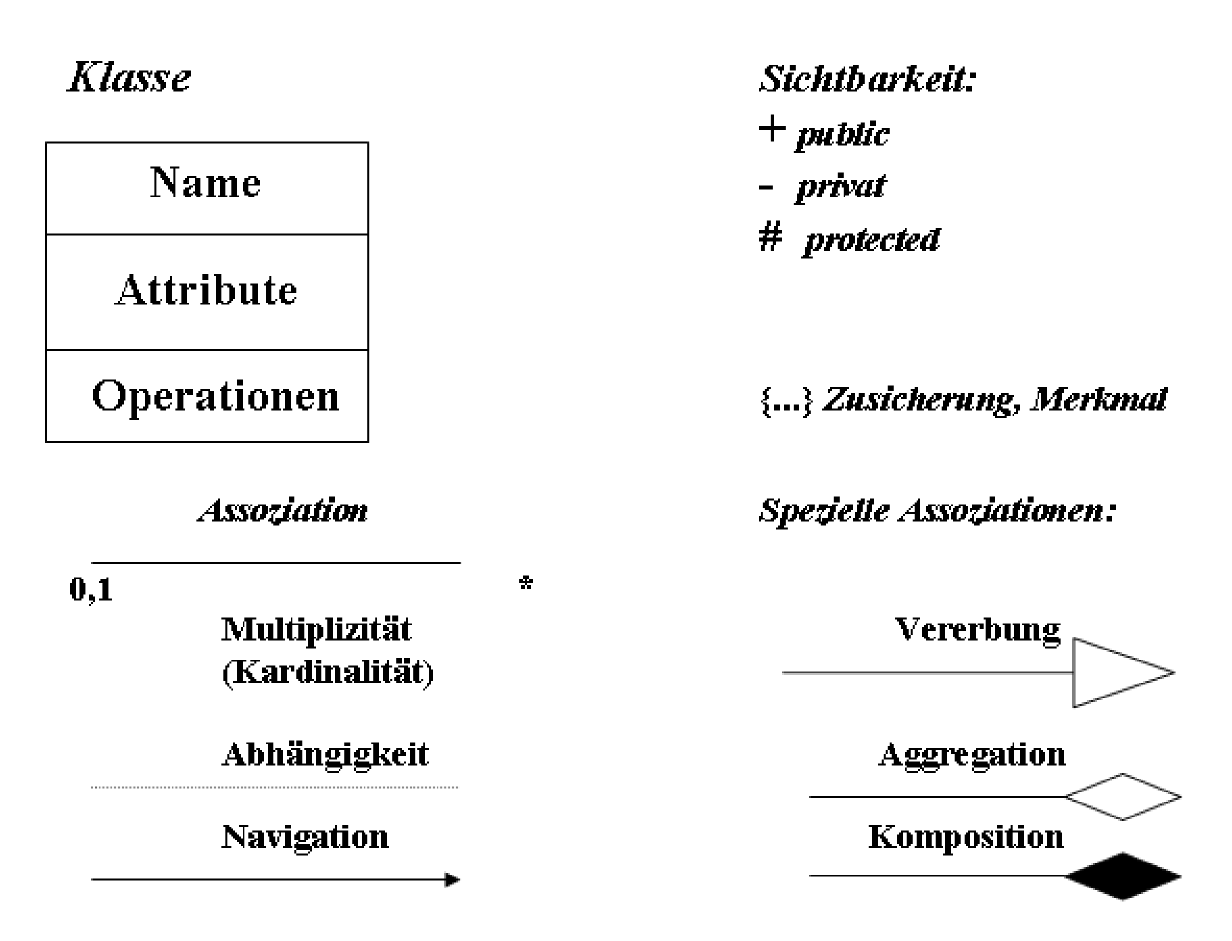


Abbildung 31: Kurzübersicht Elemente des UML Klassendiagramms

Schreibweise von Attributen:
Attribute beginnen mit einem Kleinbuchstaben.

```
Sichtbarkeit name : Typ = Initialwert {Merkmal}{Zusicherung}

# temperatur : uint8_t = 25  {temeratur > 0}
```

Schreibweise von Operationen:
Operationen beginnen mit einem Kleinbuchstaben.

```
Sichtbarkeit name (Parameter:Typ = Standardwert ...) : Rückgabe-
typ {Merkmal}{Zusicherung}

+ setTemperatur ( t : integer = 25 ) : BOOL {t > 0}
```

11.2 Aufgabenstellung

Um die Arbeitsweise und den Umgang mit dem Klassendiagramm in SiSy zu erlernen, realisieren Sie bitte das folgende einfache Beispiel.

Es soll eine Mikrocontrollerlösung entwickelt werden, bei der auf Tastendruck eine LED eingeschaltet wird. Dabei soll der Taster an Port B 0 und die LED an Port B 1 angeschlossen werden

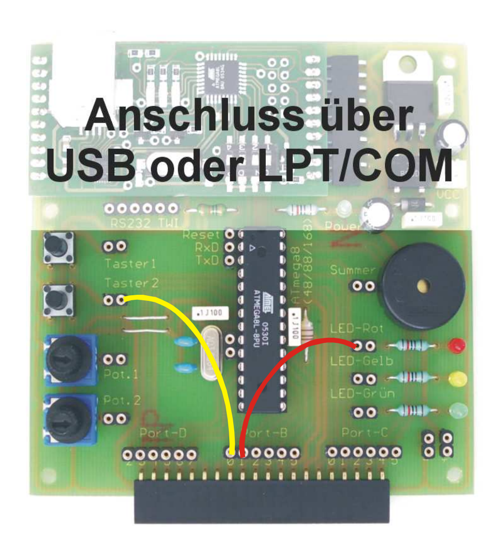
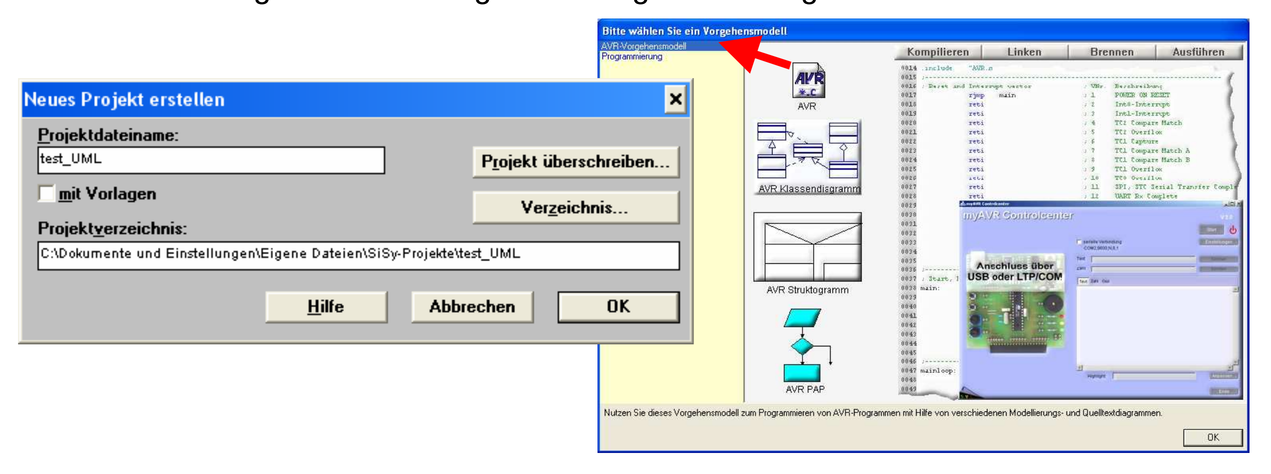


Abbildung 32: UML Beispiel

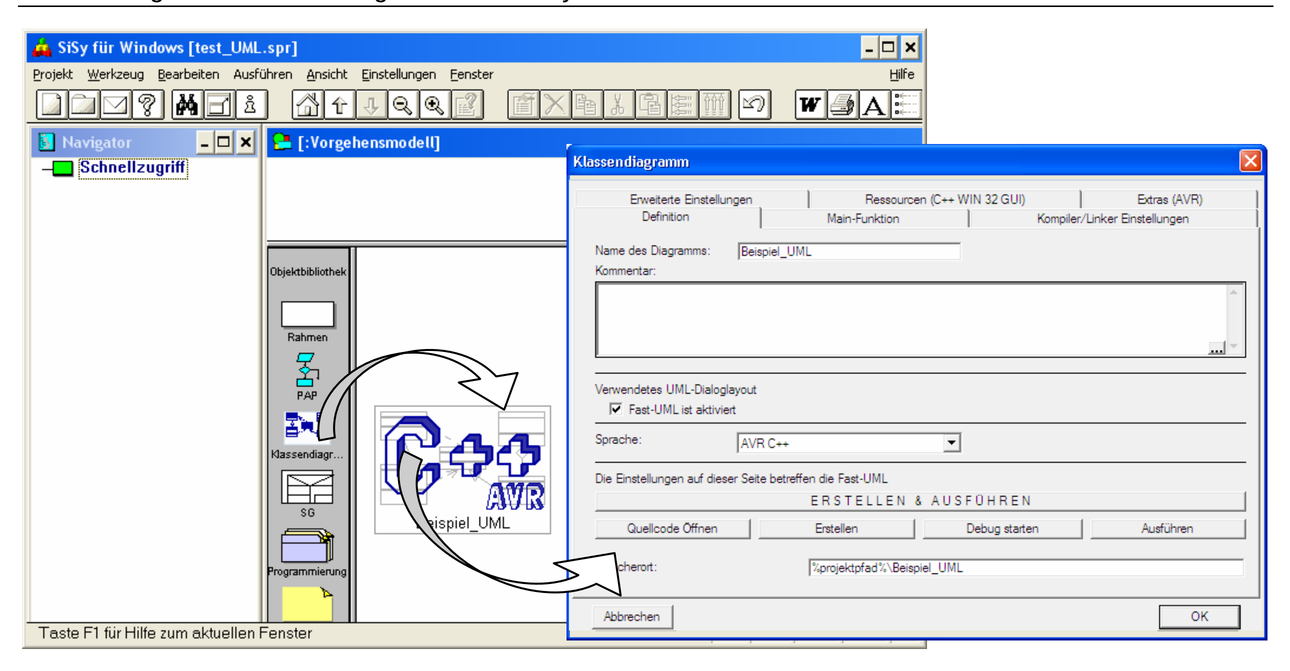
11.3 Vorbereitung

Starten Sie ggf. SiSy.

Erstellen Sie ein neues SiSy-Projekt (Hauptmenu -> Projekt -> Neu) mit dem Namen „Test_UML“. Wählen Sie das Vorgehensmodell „AVR-Vorgehensmodell“. Danach erscheint ein Dialogfenster mit möglichen Diagrammvorlagen.



Legen Sie im weiteren Verlauf die AVR-Grundeinstellungen für das Projekt fest. Ziehen Sie ein Objekt vom Typ „Klassendiagramm“ per Drag & Drop in das Diagrammfenster. Legen Sie den Namen für das Klassendiagramm fest.



11.4 Grundstruktur laden

Um das Klassenmodell zu entwickeln, muss das Klassendiagramm geöffnet werden. Wählen Sie auf dem Objekt rechte Maustaste -> Kontextmenü -> nach unten (öffnen).

Beispiel_UML

Nach unten (öffnen)
Definieren...
in neuem Fenster öffnen
Hierarchie öffnen
Löschen
Layersteuerung
Jo-Jo
Marker
Objekttyp...
aus Rahmen entfernen
Verknüpfung mit Datei
Tipps

Diagrammvorlagen
Grundgerüst (GG KC1101)
LED an (BSP KC2101)
Blinklicht (BSP KC2102)
Lauflicht (BSP KC2103)
Taster (BSP KC2104)
Signalton (BSP KC2105)
keine Vorlage verwenden
den Assistenten benutzen
Info
Abbrechen
OK

Controller

+ main() : void
+ run() : void

```
// hier Initialisierung
// ...
run();
```

```
do {
// hier
// Eingabe
// Verarbeitung
// Ausgabe
} while (true);
```

Die Struktur einer objektorientierten Mikrocontrollerlösung in SiSy erfordert im Klassenmodell eine Applikationsklasse (Hauptklasse), die sich dadurch auszeichnet, dass diese über eine Methode (Operation) mit dem Namen **main** verfügt. Der Codegenerator erzeugt eine Instanz diese Klasse und ruft die Main-Methode auf.

```
#define cpp_Test_UML
#define F_CPU 3686400
#include <io.h>
#include "Controller.cpp"

main (void)
{
    Controller MyApp;
    MyApp.main();
}
```

11.5 Systemstruktur entwerfen

Die Systemstruktur einer objektorientierten Anwendung bildet die Objekte und deren Beziehungen im Programm ab, welche im realen System als physische Elemente vorhanden sind. Als Bauplan der Objekte dienen Klassendeklarationen, welche die Eigenschaften (Attribute) und das Verhalten (Methoden/Operationen) der Objekte beschreiben. Das Klassendiagramm beschreibt also die Struktur der Klassen (Baupläne der Objekte) und die Beziehungen zwischen den Klassen. In unserer Aufgabenstellung finden wir die Objekte des Systems als **Substantive**, deren Beziehungen und Verhalten als **Verbalphrasen**.

*Es soll eine Mikro**controller**lösung entwickelt werden, bei der durch **drücken** eines **Tasters** eine **LED** **eingeschaltet** wird. Dabei soll der Taster an Port B 0 und die LED an Port B 1 **angeschlossen** werden.*

Daraus lässt sich folgende Klassenstruktur ableiten:

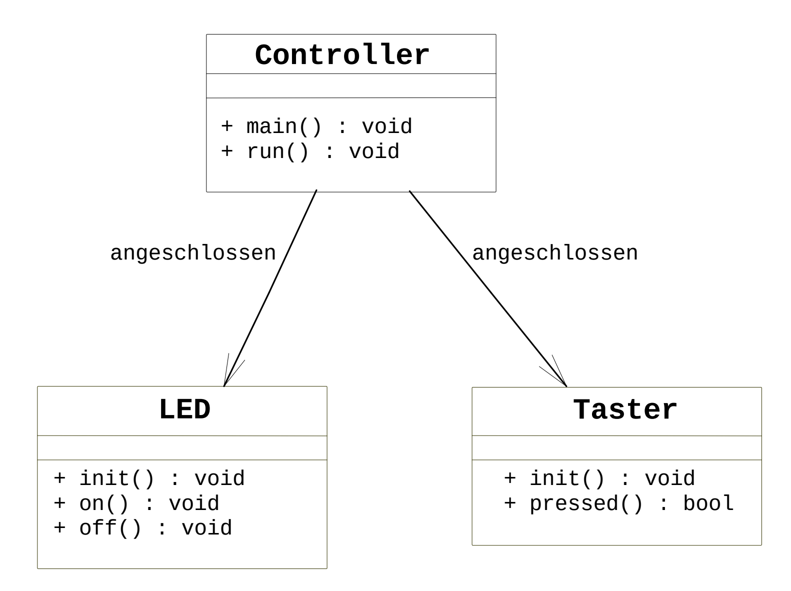
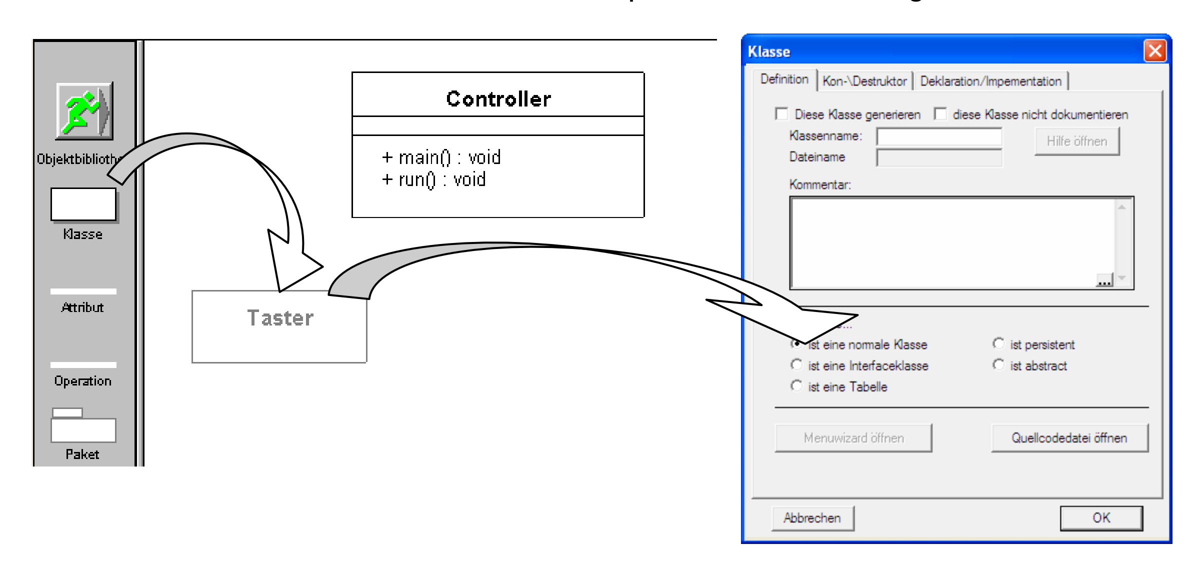


Abbildung 33: Klassenstruktur, Systementwurf mit dem UML Klassendiagramm

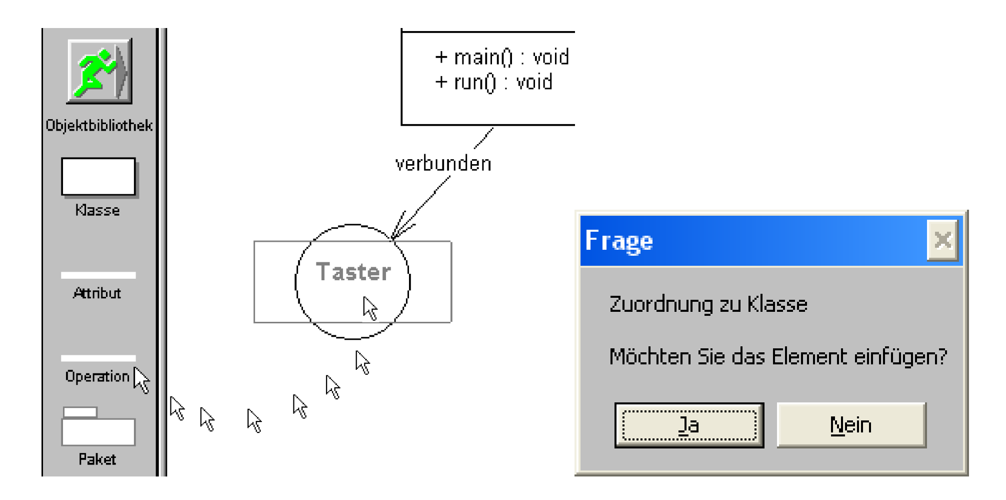
Zum Erstellen dieses Klassenmodells sind folgende Arbeitsschritte nötig:

1. **Klassen einfügen und definieren**, ziehen Sie dazu das Objekt vom Typ „Klasse“ per Drag & Drop aus der Objektbibliothek in das Diagramm. Definieren Sie den Namen der Klasse und setzen die Option „diese Klasse generieren“.

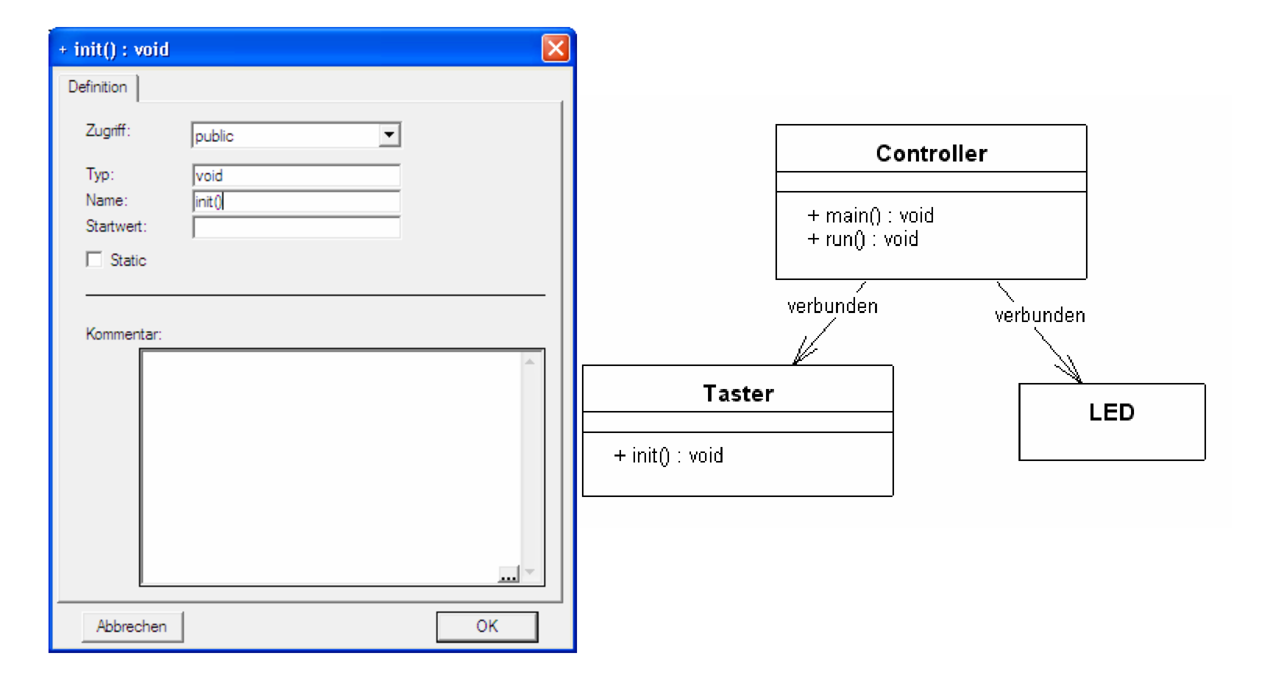


2. **Klassen verbinden**, selektieren Sie die Klasse, von der aus eine Verbindung gezogen werden soll. Ziehen Sie per Drag & Drop ausgehend vom roten Verteiler eine Verbindung auf die gewünschte Klasse. Wählen Sie den Verbindungstyp, zum Beispiel „Assoziation gerichtet“ und beschriften Sie die Verbindung.

3. **Operationen und Attribute einfügen und definieren**, ziehen Sie dazu ein Objekt vom Typ „Operation“ oder „Attribut“ aus der Objektbibliothek auf die Klasse, in die das Attribut oder die Operation eingefügt werden soll. Bestätigen Sie die Sicherheitsabfrage zum Einfügen des Elementes.



Definieren Sie Zugriff (Sichtbarkeit), Name, Typ und Parameter der Operation bzw. Zugriff (Sichtbarkeit), Name, Typ und Initialwert des Attributes.



Vervollständigen Sie das Klassenmodell entsprechend der Abbildung 34.

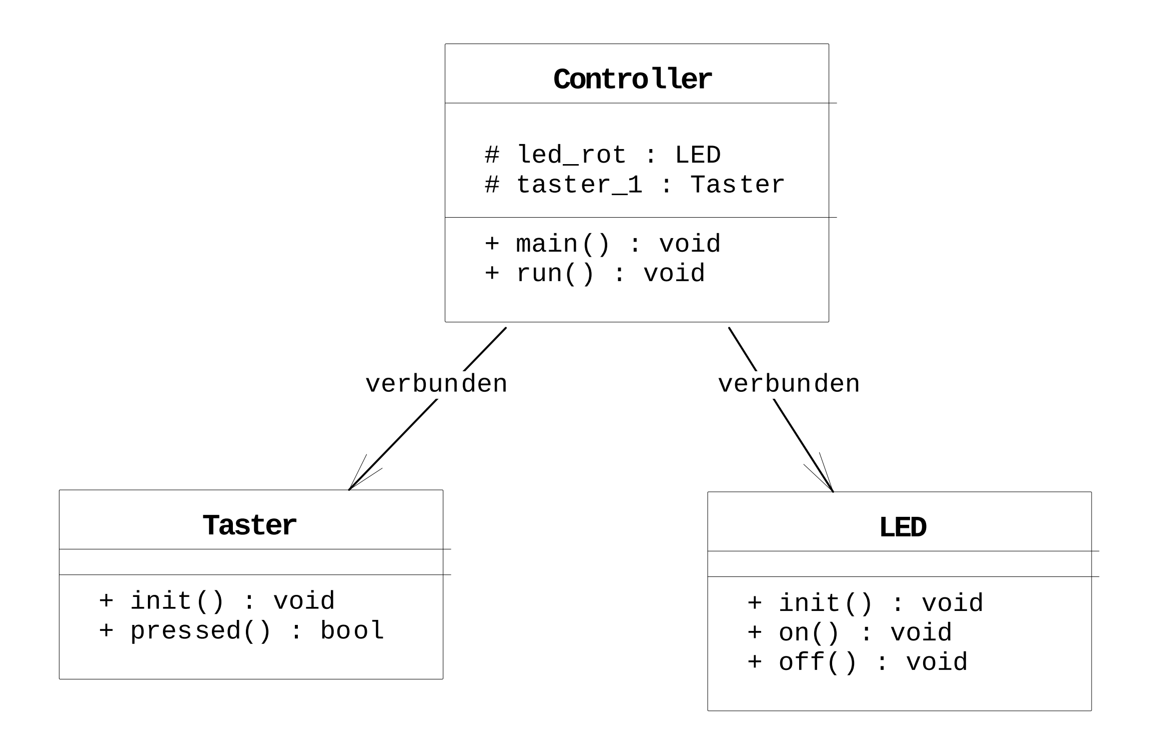
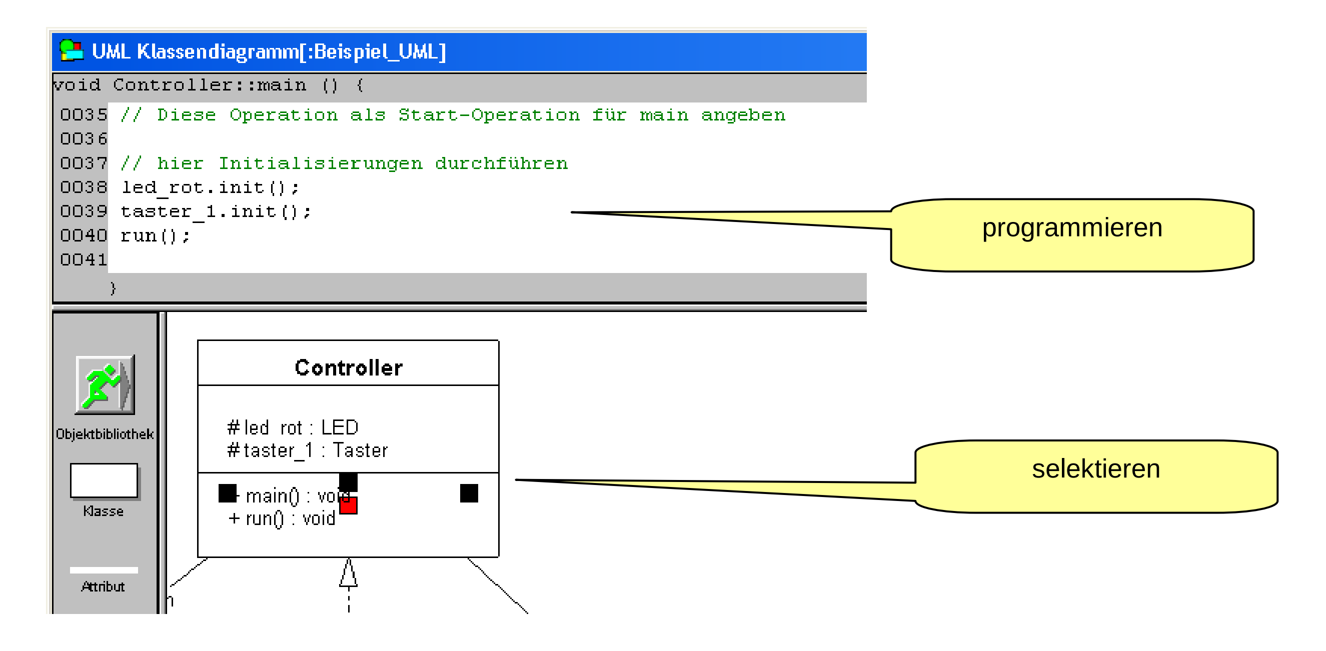


Abbildung 34: endgültiger Systementwurf mit dem UML Klassendiagramm

11.6 Systemverhalten programmieren

Die Operationen müssen mit der entsprechenden Logik versehen werden. Sie können den Quellcode der Operationen bearbeiten, indem Sie die gewünschte Operation selektieren und im Beschreibungs- / Quellcodefester die Befehle eingeben.



```
led_rot.init();
taster_1.init();
run();
```

Controller

\# led_rot : LED
\# taster_1 : Taster

\+ main() : void
\+ run() : void

```
do {
   if (taster_1.pressed())
      led_rot.on();
   else
      led_rot.off();
} while (true);
```

verbunden

verbunden

Taster

\+ init() : void
\+ pressed() : bool

LED

\+ init() : void
\+ on() : void
\+ off() : void

```
sbi(DDRB,1);     //Ausgang
off();           //LED off
```

```
cbi(DDRB,0);        //Eingang
sbi(PORTB,0);       //PullUp
```

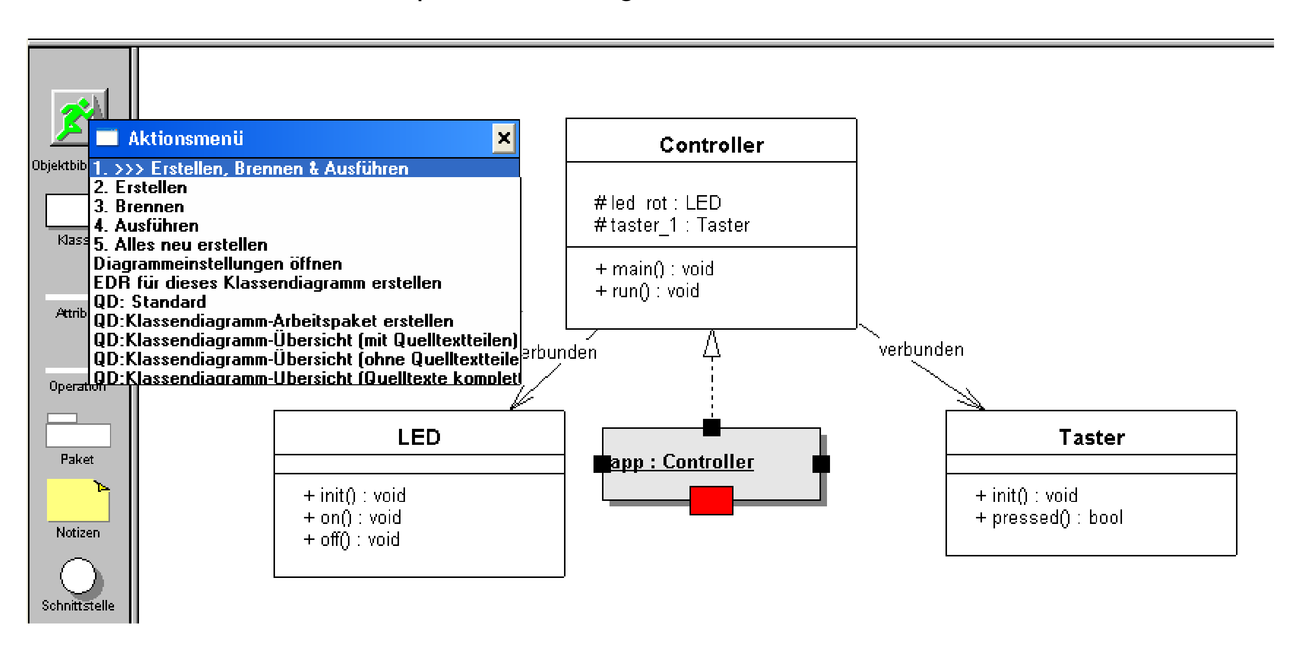
```
sbi(PORTB,1);   //LED on
```

```
if (!(PINB&0x01))
      return true;
else
      return false;
```

```
cbi(PORTB,1);   //LED off
```

11.7 Übersetzen, Brennen und Testen

Die Codegenerierung aus dem UML Klassendiagramm, das Kompilieren, Linken und Brennen kann über das Ampelmännchen gestartet werden.



Sie erhalten im Ausgabefenster ein Protokoll der ausgeführten Aktionen.

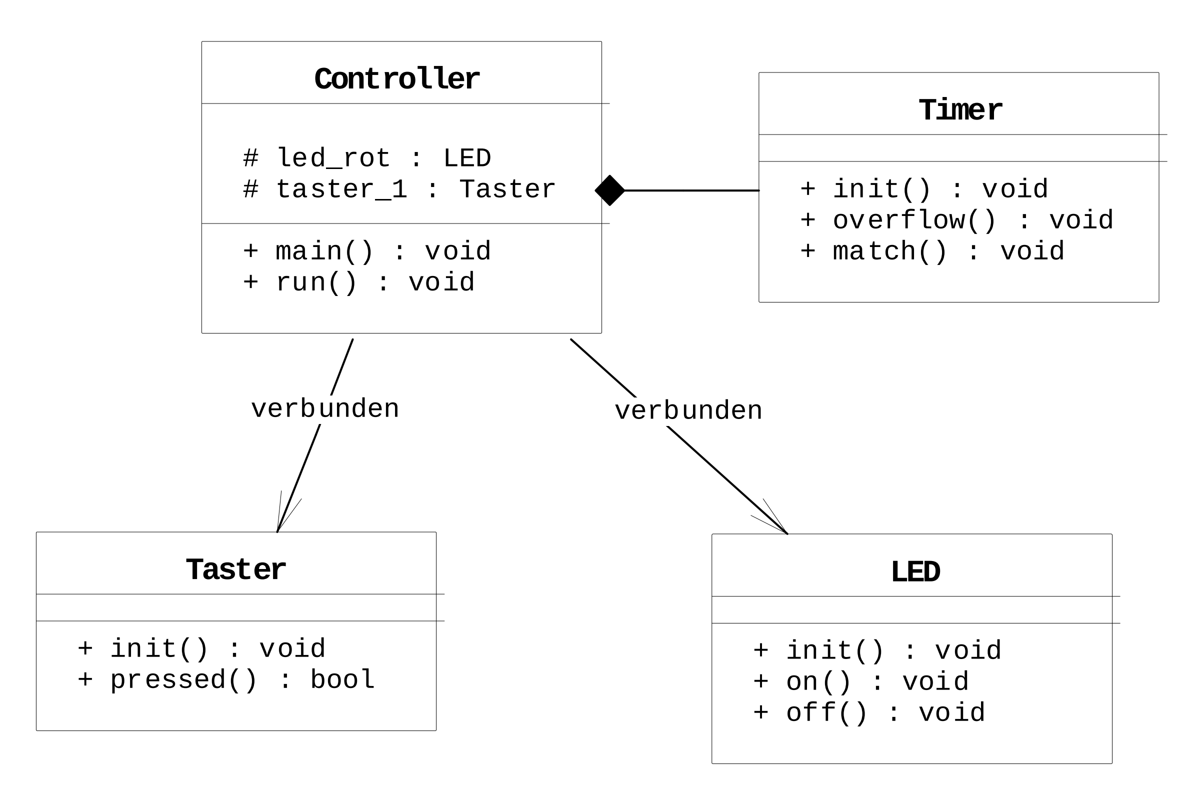
Generiere Quelltexte...
Prüfe Diagramme: 1 von 1 Paketen
Bearbeite Diagramme...
Speichere Beispiel_UML.cpp.
Speichere Controller.cpp.
Speichere Controller.h.
Speichere Taster.cpp.
Speichere Taster.h.
Speichere LED.cpp.
Speichere LED.h.
Kompiliere die Datei Beispiel_UML.cpp.
Linke Datei Beispiel_UML.elf.

Brenne Daten aus Datei Beispiel_UML.elf.
Öffne myAVR ControlCenter
Ende.

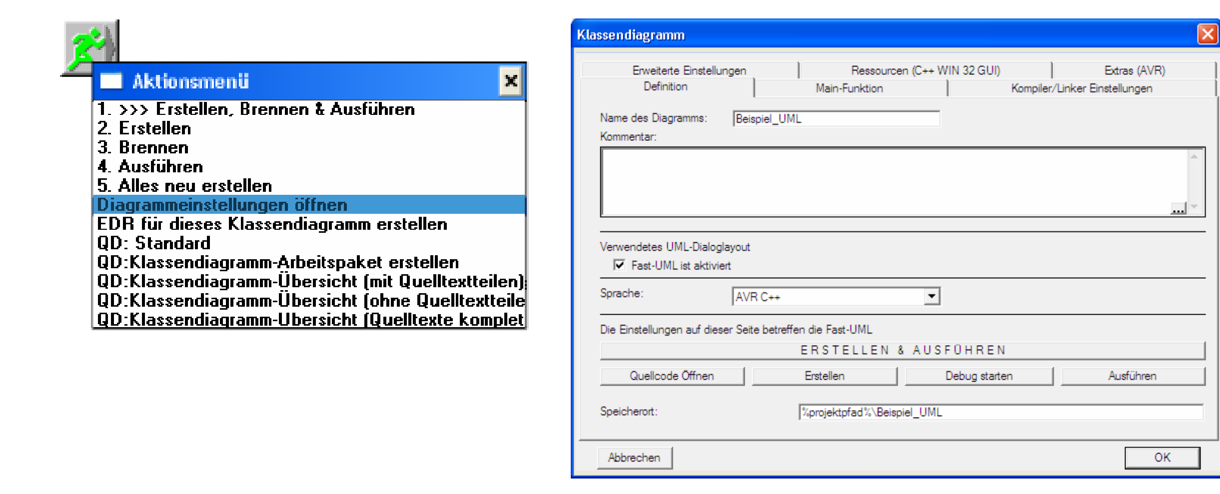
11.8 Interrupt-Service-Routinen (ISR) im Klassendiagramm

Interrupt-Service-Routinen (im weiteren ISR) werden in AVR C++ durch das Schlüsselwort ISR gekennzeichnet. Sie bilden eigenständige Funktionen. Die Besonderheit bei der Realisierung einer ISR liegt darin, dass es sich hier um ein C-Makro handelt und nicht um eine echte Funktion. Diese können also keine Methode einer Klasse sein. Um eine ISR im Klassendiagramm zu realisieren gehen Sie wie folgt vor (Erweiterung des Beispiels um die Klasser Timer und der **`ISR Timer0 Overflow`**):

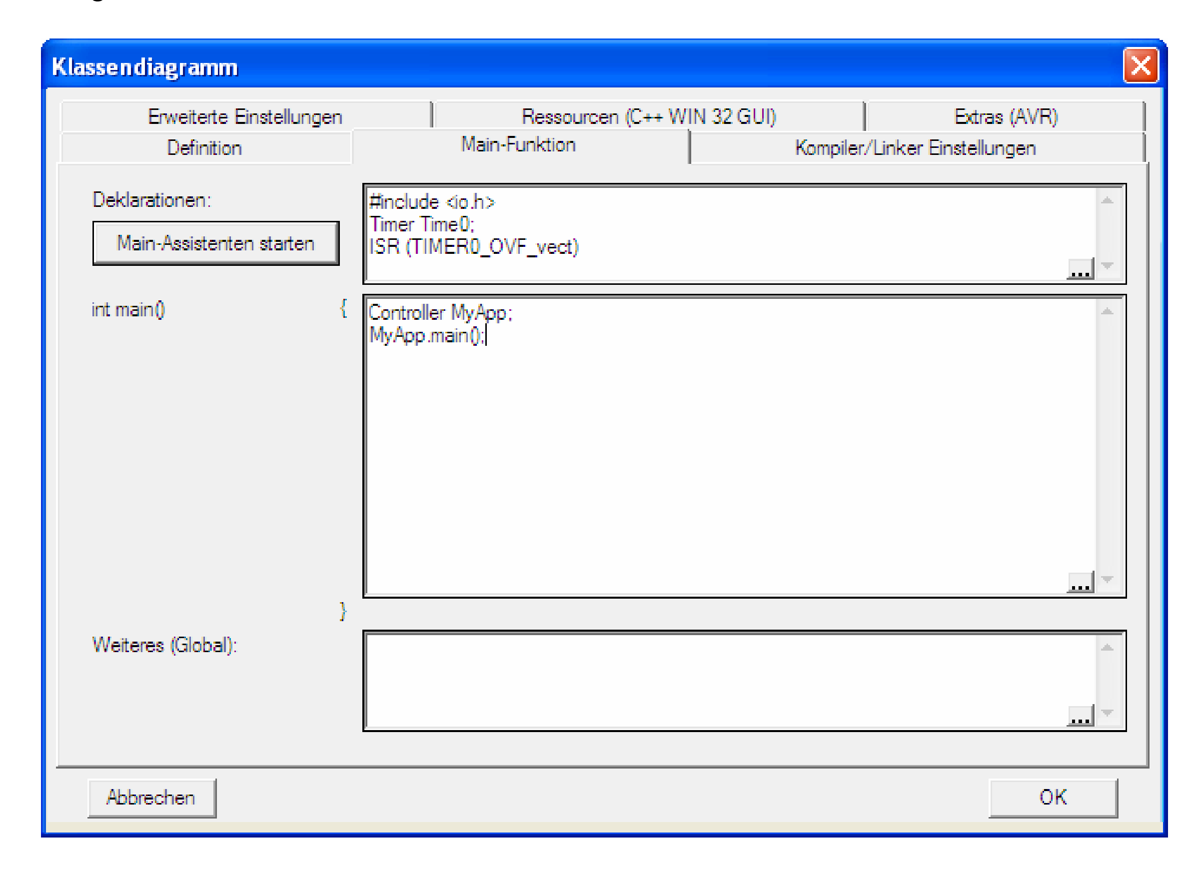
1. Die Klasse für die interruptfähige Komponente modellieren und die Methode einfügen, welche beim Interrupt ausgeführt werden soll. In unserem Beispiel ist es die Klasse **Timer** mit der Methode **overflow**. Die Initialisierung des Timers muss entsprechend erfolgen.



2. Den Dialog „Diagrammeinstellungen“ öffnen. Sie erreichen diesen Dialog über das Ampelmännchen, Menüpunkt: „Diagrammeinstellungen öffnen“



3. Wählen Sie in diesem Dialog das Dialogfeld „Main-Funktion“ und ergänzen im Eingabefeld „Deklarationen“ das ISR-Makro.



Beispiel:

```
#include  <io.h>

// Instanz der interruptfähigen Klasse anlegen
Timer Timer0;

// Interuptmakro mit aufruf der betreffenden Methode
ISR (TIMER0_OVF_vect)
{
     Timer0.overflow();
}
```

In der Interrupt-Service-Routine rufen Sie einfach die Methode der betreffenden Klasse auf.

12 Einstellungen Fuse- und Lock-Bits mit SiSy

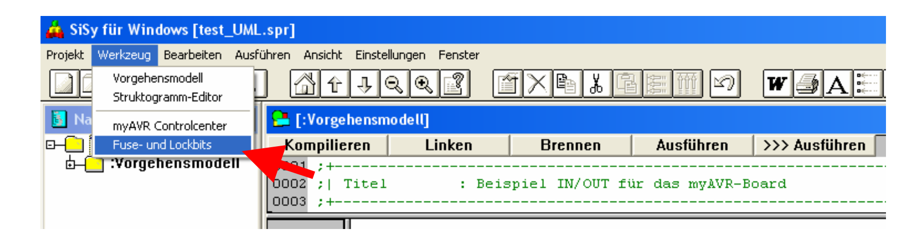
12.1 Einleitung

Fuse- und Lockbits (engl. fuse = Sicherung, engl. lock = Schloss) nennt man die Bits in bestimmten Registern des AVR zum Konfigurieren des Controllers. Die Fusebits müssen über ein entsprechendes Interface (Software) eingestellt werden. Der normale Programmiermodus verändert die Fuse- und Lockbits nicht. Je nach Controllertyp sind unterschiedliche Fuse- und Lockbits verfügbar. Die verbindliche und exakte Beschreibung findet man im jeweiligen Datenblatt des Controllers. Das falsche Setzen der Fuse- und Lockbits zählt zu den häufigsten Problemen bei der Programmierung von AVR-Controllern, daher sollte hier mit Umsicht vorgegangen werden. Das Verändern der Fusebits sollte man nicht als Anfänger vornehmen. Das geöffnete Datenblatt zur Überprüfung der Konfiguration ist das wichtigste Instrument um Fehler zu vermeiden.

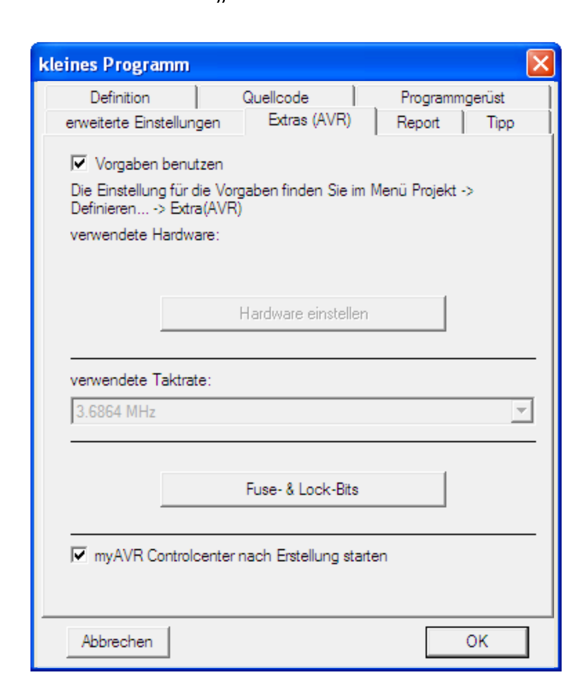
12.2 Fuse- und Lockbits, Benutzeroberfläche in SiSy AVR

Sie erreichen die Benutzeroberfläche zum Auslesen, Verändern und Programmieren der Fuse- und Lockbits in SiSy AVR wie folgt:

1. Hauptmenü -> Werkzeuge -> Fuse- und Lockbits



2. Auf Programm-Objekten wie „kleines Programm“, „Programm“, „PAP“ usw. rechte Maustaste -> Kontextmenü -> Definieren … Dialogfeld Extras (AVR) -> Schaltfläche „Fuse- & Lock-Bits“.



Beim Start der Fuse- und Lockbits-Benutzeroberfläche wird eine Verbindung zum Controller aufgebaut, der Controllertyp ermittelt, die Fuse- und Lockbit-Definitonen des Controllers geladen, die Fuse- und Lockbit Einstellungen des Controllers ausgelesen und angezeigt. Dieser Vorgang kann je nach Controllertyp und Verbindung einige Sekunden

dauern. Die Verbindung zum Controller wird, solange diese Benutzeroberfläche offen ist, dauerhaft offen gehalten.

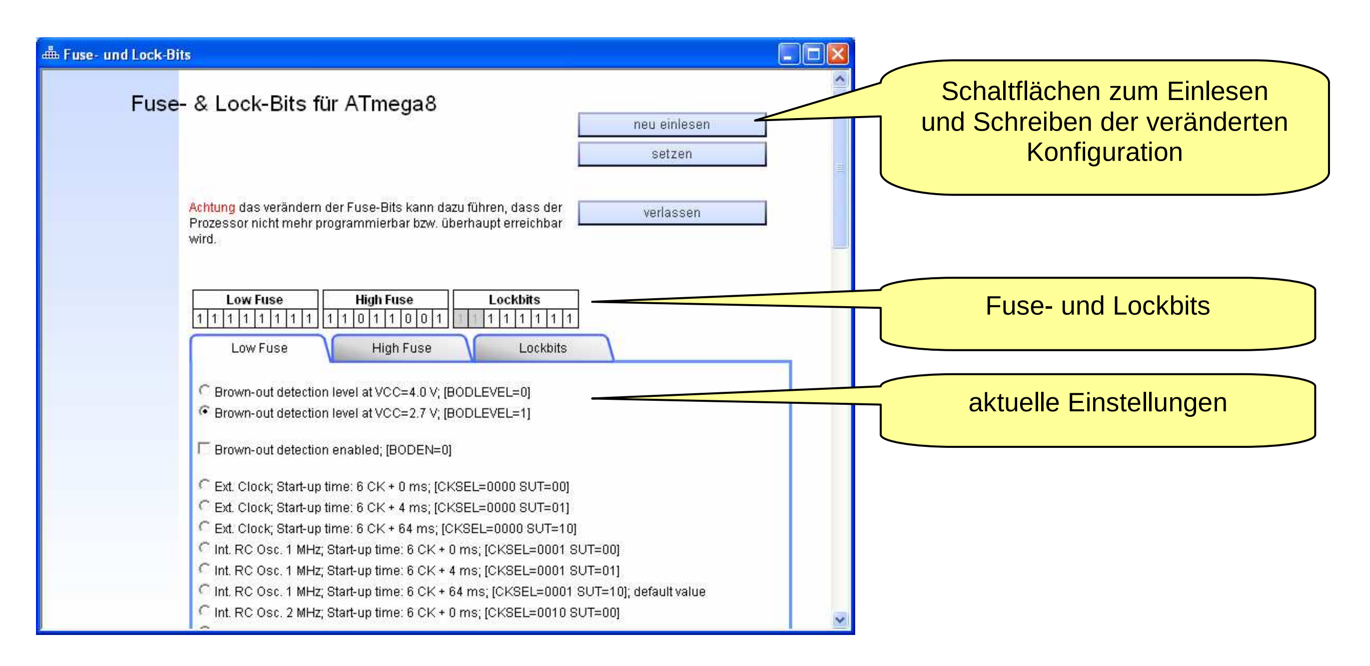


Abbildung 35: Fuse- und Lockbits des ATmega8

Die Benutzeroberfläche passt sich dem ermittelten Controller und den dazugehörigen Definitionsdaten automatisch an. Es werden immer nur die Optionen angezeigt, die zum ermittelten Controller gehören. Vergleichen Sie dazu immer das betreffende Datenblatt. Es ist nicht zulässig, den Programmer oder den Controller während der Sitzung zu entfernen oder zu wechseln. Dazu ist dieses Fenster zu schließen, danach der Controller oder Programmer zu wechseln und die Benutzeroberfläche erneut zu starten.

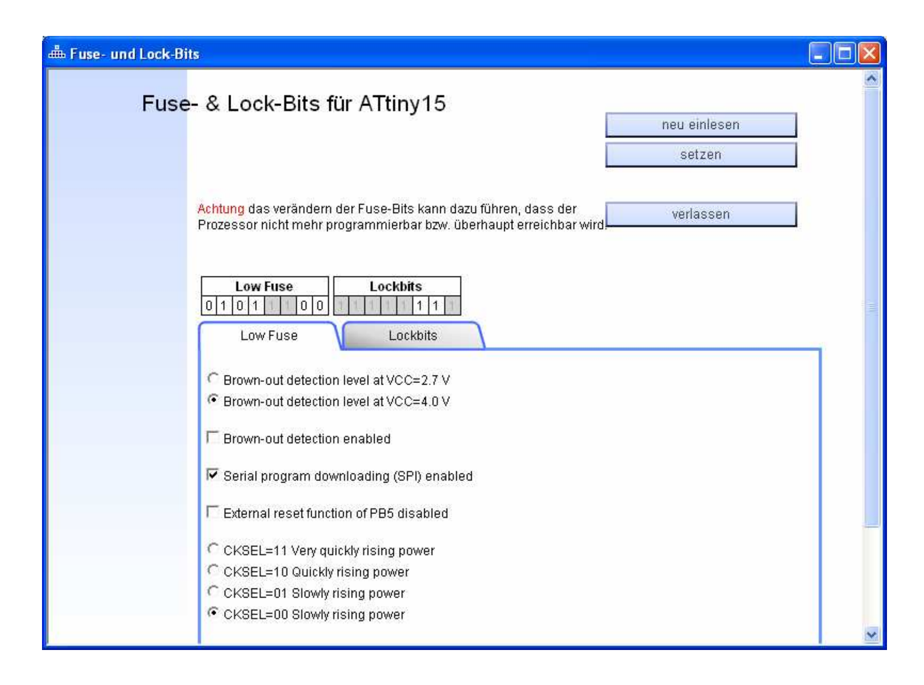


Abbildung 36: Fuse- und Lockbits des ATtiny15

12.3 Fuse- und Lockbits verändern

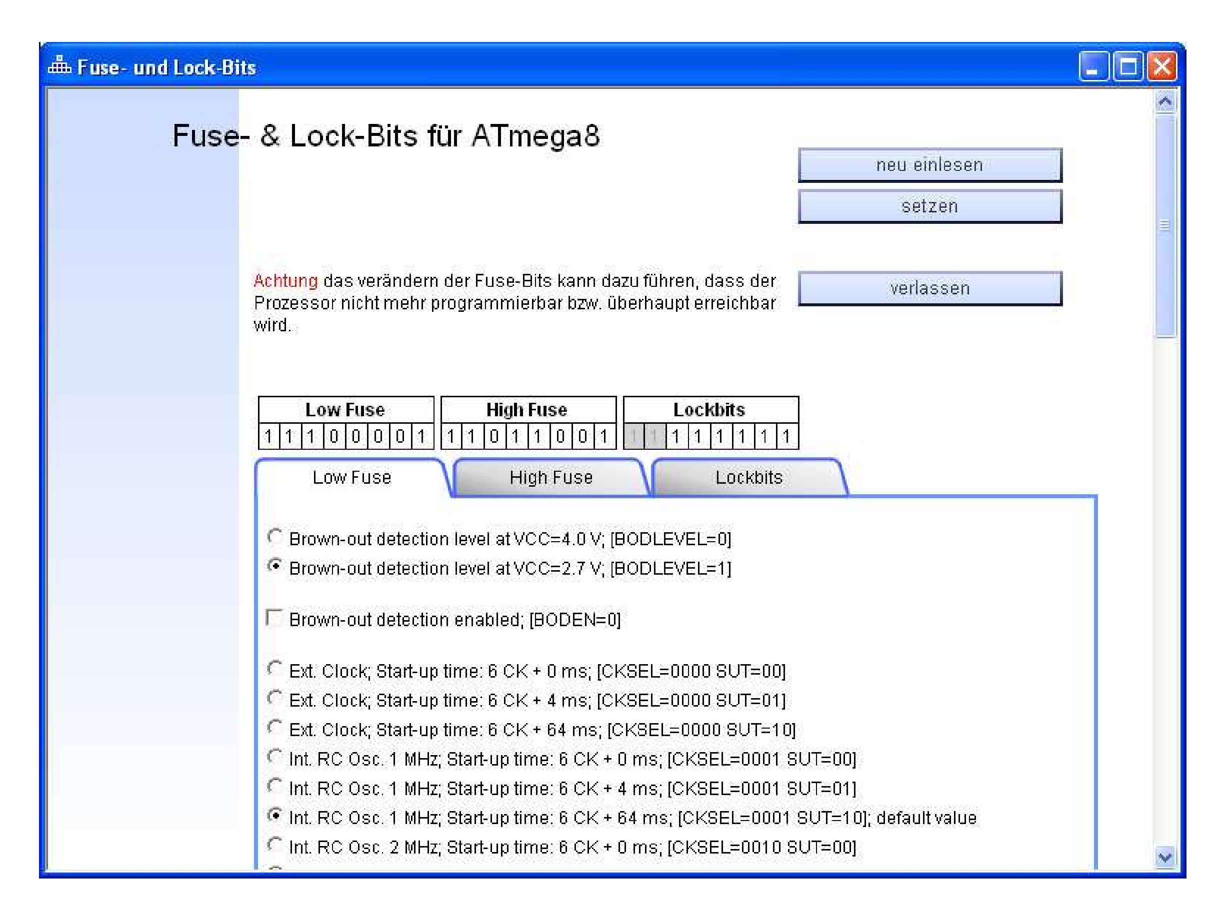
Zum Verändern der Fuse- und Lockbits sollte der entsprechende Abschnitt im Datenblatt des Controllers studiert werden. Über die Dialogfelder Low-, High- und Extended-Fuse- sowie Lockbits können die einzelnen Optionen bequem ausgewählt werden. Die Änderungen werden im Anzeigebereich für die Fuse- und Lockbits visualisiert. Erst mit dem Betätigen der Schaltfläche „setzen“ werden die neuen Einstellungen an den Controller übertragen.

Beachte: Falsche Einstellungen der Fuse- oder Lockbits können dazu führen, dass der Controller in der aktuellen Hardware nicht mehr angesprochen werden kann.
Häufige Fehleinstellungen durch unerfahrene Entwickler sind:

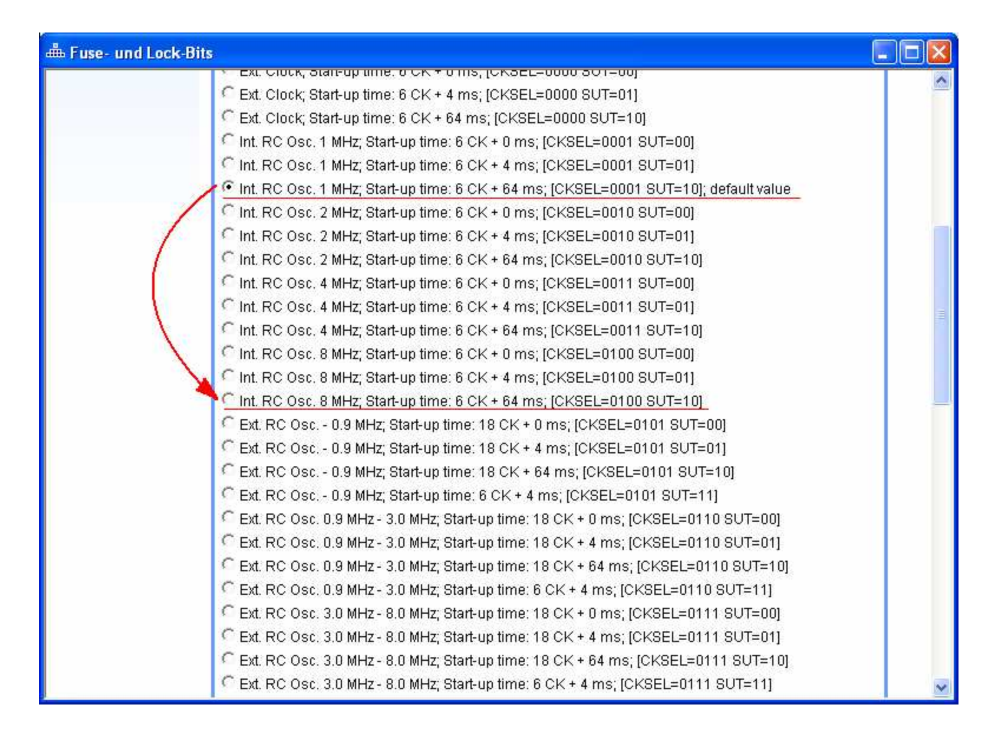
- **Reset disable ->** führt dazu, dass kein ISP mehr möglich ist
- **ISP enable ausgeschaltet ->** führt dazu, dass kein ISP mehr möglich ist
- **Taktquelle umgeschaltet ->** führt u.U. dazu, dass der Controller nicht arbeitet

Im Folgenden wird die Vorgehensweise beschrieben, wie die Taktquelle eines ATmega8 vom internen 1 MHz Oszillator auf intern 8 MHz umgeschaltet wird.

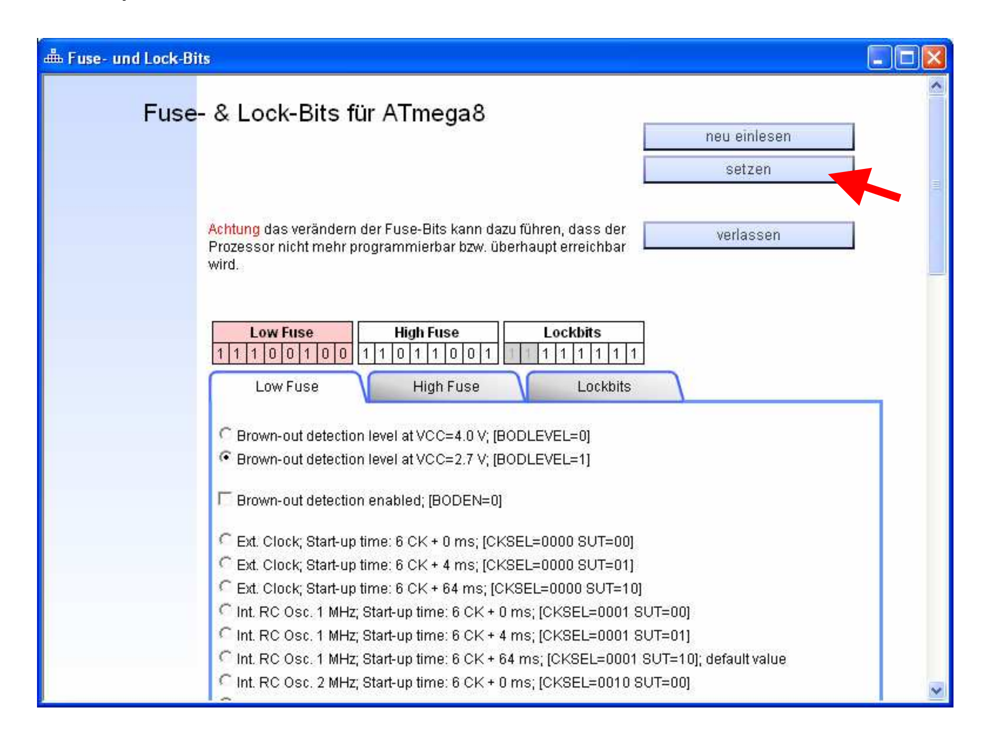
1. Das Board und den Programmer anschießen.
2. Benutzeroberfläche für das Verändern der Fuse- & Lockbits starten.



3. in der Liste der Optionen nach unten scrollen und **`Int. RC. Osc. 8MHz`** auswählen. Vergleichen Sie dazu die Beschreibung im Datenblatt des ATmega8.



4. Die Optionsseite wieder zurückscrollen und die veränderten Fusebits überprüfen.



5. Die Schaltfläche „setzen“ wählen und die Sicherheitsabfrage bestätigen. Danach sollten die Einstellungen überprüft werden durch das Betätigen der Schaltfläche „neu einlesen“.

Über die Schaltfläche „verlassen“ kann die Sitzung zum Verändern der Fuse- und Lockbits beendet werden. Die Verbindung zum Controller und Programmer wird dann geschlossen.

Anhang: Tastaturbelegung, allgemein

Die Tastenbelegung ist abhängig vom jeweiligen Diagramm und der verwendeten Ausgabe:

Taste	Funktion
F1	Hilfe wird geöffnet
F2	Zoomen
F3	Objekt wird dem Layer zugefügt
F4	Objekt wird vom Layer entfernt
F5	Farbe bei Rahmen wird geändert; Form am Anfang einer Verbindung ändert sich
F6	Bei Rahmen und Verbindungen ändert sich die Form
F7	Bei Rahmen und Verbindungen ändert sich der Linientyp
F8	Form am Ende einer Verbindung ändert sich
F9	Bewirkt, dass der Mittelpunkt einer Kante auf Null gesetzt wird
ESC	Im Diagramm: Nach oben
	Im Text-Editormodus: Beendet ihn mit Rückspeichern
	Im Dialog: Bricht ihn ab ohne zurückzuspeichern
Tab	In der Reihenfolge, in der die Objekte erstellt wurden, werden sie markiert
Umschalttaste + Enter	Objektbeschreibung wird geöffnet
Leertaste	Objektbeschreibung wird geöffnet
Alt + Enter	Dialog Definieren
Strg + Enter	Diagramm nach unten
Strg + `R	Report für selektiertes Objekt
Strg + `A`	Executebefehl ausführen (nur in bestimmten Ausgaben)
Strg + `D`	Diagrammreport wird aufgerufen
Strg + `I`	Import von Diagrammen
Strg + `T`	Tokensteuerung starten/beenden
Strg + Shift + `X`	Export von Diagrammen
Strg + `+`	Selektiertes Objekt wird vergrößert (nur in bestimmten Ausgaben)

Strg + `-`	Selektiertes Objekt wird verkleinert (nur in bestimmten Ausgaben)
Strg + `*`	Ursprüngliche Objektgröße wird wiederhergestellt
Strg + Maustaste	Selektiertes Objekt wird innerhalb des Diagramms kopiert
Strg + Cursortasten:	
- Cursor nach links	Selektiertes Objekt wird in X-Richtung verkleinert
- Cursor nach rechts	Selektiertes Objekt wird in X-Richtung vergrößert
- Cursor nach oben	Selektiertes Objekt wird in Y-Richtung vergrößert
- Cursor nach unten	Selektiertes Objekt wird in Y-Richtung verkleinert
Enter	Editormodus zum Definieren der Objekte
Entf	Löschen
Cursortasten	Selektiertes Objekt wird verschoben (in Verbindung mit der Umschalttaste sind größere Schritte möglich)
`+`	Diagramm vergrößern
`-`	Diagramm verkleinern
`*`	Einpassen des Diagramms

Anhang: Tastenbelegung im Struktogramm

F3	öffnet den Quelltext
F5	startet das Programm (bei SiSy AVR das myAVR Controlcenter)
F8	Codegenerierung des aktuellen Struktogramms
F9	übersetzt das Struktogramm
ESC	Ebene nach oben
Num `+`	selektiertes Objekt wird vergrößert
Num `-`	selektiertes Objekt wird verkleinert
Enter	ermöglicht die Bearbeitung des Quelltextes im Struktogramm-Editor
Entf	löschen
T	ermöglicht die Bearbeitung des Titels
K	ermöglicht die Bearbeitung des Kommentars
B	Begin neue Funktion einfügen
S	Sequenz einfügen (Folge/Block)
D	Do einfügen (elementare Aktion)
W	While-Schleife einfügen (kopfgesteuerte Schleife)
R	Repeat-Schleife einfügen (fußgesteuerte Schleife)
F	For-Schleife einfügen (Zählschleife)
A	Alternative einfügen
C	Case einfügen (Fallunterscheidung)
I	If-Zweig einfügen
E	Exit einfügen
L	Loop einfügen
Cursortasten	Bewegung im Struktogramm

Anhang: Mausoperationen

Die Maus hat in SiSy eine Anzahl von nützlichen Funktionen, welche die Arbeit in Projekten erleichtern.

Selektion	*Klick auf Objekt* Objekt ist markiert und kann separat weiterbearbeitet werden.
Selektion aufheben	*Klick auf Fensterhintergrund* Aufhebung der Objektmarkierung.
Mehrfachselektion	*Umschalttaste + Klick auf Objekt* Selektion/Markierung von mehreren Objekten zur Weiterbearbeitung. *Markise* Mit gedrückter linker Maustaste auf Fensterhintergrund und Ziehen eines Rechtecks über zu markierende Objekte.
Verschieben	*Drag & Drop im Diagramm* Objekt mit linker Maustaste anfassen und verschieben. Objekte werden am Raster verschoben. *Umschalttaste + Drag & Drop im Diagramm* Verschieben von Objekten ohne Raster.
Fensterinhalt schieben	*Linke und rechte Maustaste drücken + Verschieben der Maus im Diagramm* Der komplette Diagramminhalt wird geschoben.
Objekt kopieren	*STRG + Drag & Drop* Maustaste gedrückt halten und Mauszeiger vom Objekt auf den Fensterhintergrund führen. Eine Originalkopie des Objektes wird im aktuellen oder in einem anderen Diagramm erzeugt.
Referenz erzeugen	*Drag & Drop aus Navigator* Ziehen des gewünschten Objektes aus dem Navigator in das Diagramm. Es wird eine Referenz des gewählten Objektes erzeugt. *Drag & Drop aus Objektbibliothek* Rot beschriftete Objekte können nur als Referenz erzeugt werden. Eine Liste zur Auswahl des gewünschten Typs erscheint. *Strg + Drag & Drop aus Objektbibliothek* Eine Liste zur Auswahl der gewünschten Referenz des Originalobjektes erscheint. *Drag & Drop aus anderem Diagramm* Ziehen des gewünschten Objektes aus dem Quelldiagramm in das Zieldiagramm. Es wird eine Referenz des gewählten Objektes erzeugt.

Objekt anlegen	*Drag & Drop aus Objektbibliothek* Ein Objekt aus der Objektbibliothek wird im Diagramm angelegt und steht zur Verfeinerung bereit.
Objekt anhängen	*Drag & Drop Verteiler auf Fensterhintergrund* Durch Ziehen einer Kante vom Verteiler auf den Fensterhintergrund wird ein neues Objekt erzeugt. Nach Auswahl des Objekttyps sind die Objekte miteinander verbunden.
Objekte verbinden	*Drag & Drop Verteiler zu Objekt* Klick auf den Verteiler des zu verbindenden Objektes. Bei gedrückter linker Maustaste auf das gewählte Objekt ziehen. *Verbindung aus Objektbibliothek (in der UML)* Hierbei wird erst die gewünschte Verbindung in der Objektbibliothek angeklickt und danach die beiden zu verbindenden Objekte im Diagramm nacheinander.
Verbindung anordnen	*Drag & Drop Mittelpunkt* Beliebige Gestaltung der Verbindung durch Ziehen mit der Maus.
Verbindung ändern	*Drag & Drop Anfangs-/Endpunkt einer Kante* Für die Verbindung wird ein neues Zielobjekt gewählt.
Objekt definieren	*Doppelklick auf Objekt* Durch Doppelklick auf Objekte öffnet sich das Kontextmenü. Bei Abschalten des Menüs unter *Einstellungen/Menü bei Doppelklick* erscheint eine Zeile zur Namensgebung. Mit ESC wird die Eingabe bestätigt. *Doppelklick auf Verteiler* Es wird der Definieren-Dialog aufgerufen, in dem das Objekt benannt und beschrieben werden kann.
Kontextmenü öffnen	*Klick mit rechter Maustaste auf Objekt*
Fenster neu zeichnen	*Doppelklick auf Fensterhintergrund* *Hinweis*: Doppelklick mit linker Maustaste wirkt wie Enter.
Fenster aktualisieren	*Strg + Doppelklick auf Fensterhintergrund* Die vom Programm ausgeführten aber noch nicht sichtbargemachten Befehle werden im Fenster erstellt. Das Fenster wird aktualisiert.